IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151156

ISL-HandNet: A Real-Time Indian Sign Language Recognition System Using MediaPipe Landmarks and a Random Forest Classifier

Jaithraa Madhav M[1], Deepak K[2], Prataparao Charith[3], Dr. G. Paavai Anand[4]

Student, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Vadapalani Campus, Chennai, India[1-3]

Assistant Professor [SG], Department of Computer Science and Engineering,

SRM Institute of Science and Technology, Vadapalani Campus, Chennai, India[4]

Abstract: Sign language is the primary means of communication for millions of deaf and hard-of-hearing people in India, yet very few hearing people are fluent in Indian Sign Language (ISL). Vision-based translators that classify raw RGB frames with deep convolutional networks are sensitive to background clutter, lighting changes and skin-tone variation, and usually need GPU hardware. This paper presents ISL-HandNet, a lightweight, real-time recognition pipeline for the 26-letter ISL finger-spelling alphabet that replaces pixel-level input with normalized skeletal landmarks. MediaPipe Hands extracts 21 three-dimensional hand keypoints per frame, producing a 63-dimensional feature vector. Translation invariance is obtained by expressing every landmark relative to the wrist, and scale invariance by dividing by the wrist-to-middle-finger-MCP distance. An ensemble Random Forest of 100 decision trees is trained on 17,589 valid landmark vectors extracted from 38,612 raw frames. On a held-out set of 3,226 frames the model reaches an overall accuracy of 87.10% (macro F1 of 0.86) with sub-millisecond classifier inference on a standard CPU. The system is verified with an automated test script and deployed as a live OpenCV webcam application. The results show that landmark-based feature engineering is a practical, low-cost basis for assistive ISL communication tools.

Keywords: Indian Sign Language; Hand Landmarks; MediaPipe Hands; Random Forest; Gesture Recognition; Computer Vision; Assistive Technology

1. INTRODUCTION

Sign language is the main medium of expression for millions of deaf and hard-of-hearing individuals across India. Because most of the hearing population does not know Indian Sign Language (ISL), communication barriers arise in classrooms, hospitals, banks, public offices and workplaces. Automated systems that convert visual hand gestures into text or speech can narrow this gap and give signers more independence in everyday interactions.

Early vision-based recognition systems relied on deep Convolutional Neural Networks (CNNs) that operate directly on raw RGB pixels [9], [10]. Pixel-level networks learn useful spatial hierarchies, but they are highly sensitive to ambient lighting, background noise, motion blur and the signer's skin tone. They also process high-dimensional image tensors, which demands GPU acceleration and makes deployment on low-cost laptops and edge devices difficult. This paper presents ISL-HandNet, a landmark-driven alternative. Instead of classifying images, the system uses MediaPipe Hands [1], [3] to extract 21 joint positions in 3D space and condenses each frame into a compact 63-element numeric array. After normalization relative to the wrist, the array describes only the geometric posture of the hand and is independent of the background and of the hand's distance from the camera.

1.1 Background and Motivation

Assistive technology for the deaf community has traditionally depended on human interpreters, whose availability is limited and whose services are costly. Recent progress in computer vision and machine learning makes it feasible to build interpreters that run on an ordinary laptop with a webcam. Finger-spelling is the natural starting point for such a system: it covers the full alphabet, it is used to spell names and technical terms that have no dedicated sign, and it forms the foundation on which word-level and sentence-level recognition can later be built.

Landmark-based recognition fits this goal well. A hand skeleton contains the information that distinguishes one handshape from another (finger flexion, thumb position, relative fingertip distances) while discarding irrelevant

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151156

appearance details. Compact skeleton features also permit lightweight classical classifiers that train in seconds and predict in well under a millisecond, which is essential for smooth real-time feedback.

1.2 Problem Statement

Many published sign recognition systems either require specialised hardware such as sensor gloves, or depend on heavy deep-learning models trained on raw images. Both choices limit practical adoption. In addition, most prior work targets American Sign Language, and far fewer lightweight systems are validated specifically for the ISL alphabet on commodity CPU hardware.

The problem addressed in this work is the design of a vision-based system that (i) recognises the 26 static ISL alphabet signs from a standard webcam, (ii) is robust to background, lighting and camera-distance changes, (iii) runs in real time on a CPU without GPU support, and (iv) is accompanied by a reproducible evaluation protocol with per-class metrics.

1.3 Technical Challenges

- **Inter-class similarity:** Several alphabet signs differ only by small changes in finger curl or thumb placement. A classifier must resolve these subtle geometric differences from noisy 3D landmark estimates.
- **Sensitivity to environmental conditions:** Lighting, shadows, clothing and cluttered backgrounds degrade pixel-based models. The pipeline must isolate the hand posture from these effects.
- **Translation and scale variance:** The same sign appears at different image positions and sizes depending on where the signer sits relative to the camera. Features must be invariant to both.
- **Hand detection reliability:** The landmark detector does not succeed on every frame, particularly with blur, occlusion or unusual hand orientations. Frames without a detected hand cannot be used for training and must be filtered out.
- **Real-time processing constraints:** Capture, landmark extraction, normalization, classification and on-screen overlay must complete fast enough on a CPU to give the user immediate feedback.

1.4 Contributions

The main contributions of this work are as follows. First, a spatial landmark vectorization pipeline converts variable RGB input into a normalized 63-dimensional feature space (21 joints × [x, y, z]) that is invariant to translation and scale. Second, an ensemble Random Forest classifier (isl_rf.joblib) is trained on 17,589 valid landmark vectors and achieves 87.10% held-out accuracy across 26 ISL classes with sub-millisecond classifier latency. Third, the approach suppresses environmental noise by discarding all non-skeletal image content. Finally, an automated verification script (test_pipeline.py) and an interactive real-time webcam application (webcam_demo.py) make the system reproducible and demonstrable.

1.5 Organization of the Paper

Section 2 reviews related work. Section 3 describes the system architecture and methodology. Section 4 presents the source code of the main modules. Section 5 reports the experimental setup and results. Section 6 discusses findings, limitations and future work, and Section 7 concludes the paper.

2. RELATED WORK

Automated sign language recognition lies at the intersection of computer vision, machine learning and human–computer interaction. Existing approaches can be grouped into sensor-based systems, handcrafted-feature vision systems, deep-learning image classifiers, and landmark-based pipelines. This section reviews each group and identifies the gap addressed by ISL-HandNet.

2.1 Sensor-Based (Glove) Methods

Sensor-based systems use data gloves fitted with flex sensors, accelerometers and gyroscopes to measure finger bending and hand orientation directly. They provide precise joint-angle information that does not depend on lighting or camera placement. However, they are intrusive, expensive, need calibration and cumbersome hardware, and restrict the natural movement of the signer. These drawbacks make them unsuitable for casual, large-scale use by the general public.

2.2 Handcrafted Features and Classical Machine Learning

The limitations of gloves motivated camera-based recognition. Early vision systems extracted handcrafted descriptors such as Scale-Invariant Feature Transform (SIFT) keypoints [7] and Histograms of Oriented Gradients (HOG) [6], and fed them to classical classifiers such as Support Vector Machines [8] or K-Nearest Neighbours. Designing these features required expert knowledge of which visual cues characterise a gesture (edge orientations, colour blobs, contours).

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151156

Performance was acceptable in controlled conditions, but such systems generalised poorly across backgrounds, lighting conditions and signers.

2.3 Deep Learning and Convolutional Neural Networks

Deep learning, and CNNs in particular, changed visual recognition by learning features automatically from data [9], [10]. Convolutional layers extract edges, textures and shapes with learnable filters; pooling layers reduce spatial dimensions and add a degree of translation tolerance; and fully connected layers combine the learned features into class scores. Deeper and more efficient architectures such as ResNet [11] and MobileNetV2 [12] have further improved accuracy and efficiency on image benchmarks.

Applied directly to sign language video, however, pixel-based CNNs have notable drawbacks. They may overfit to background textures, shadows or clothing patterns instead of hand geometry; they require large, diverse, well-labelled image datasets to generalise; and training and inference of deep networks usually require GPU resources. These issues reduce their suitability for low-cost, CPU-bound deployments.

2.4 Landmark-Based Recognition with MediaPipe

To avoid pixel-level noise, recent work first extracts a skeletal representation of the hand and classifies that instead. MediaPipe [1] is a framework for building cross-platform perception pipelines. Its Hands solution [3] combines a palm detector with a hand-landmark regression model that infers 21 three-dimensional keypoints per hand and runs in real time on mobile and desktop CPUs. Because the output is a small numeric vector, it can be classified by lightweight models, and the representation is already independent of the background, clothing and lighting.

Random Forests [2] are a natural fit for such features. A Random Forest is an ensemble of decision trees trained on bootstrap samples with random feature subsets; the final label is decided by majority vote. This reduces variance compared with a single tree, handles multi-class problems directly, needs little hyper-parameter tuning, and is fast at prediction time. Implementations are readily available in scikit-learn [4].

2.5 Sequence Models for Dynamic Signs

Many words and phrases in sign language involve motion and therefore require temporal modelling. Recurrent architectures, particularly Long Short-Term Memory (LSTM) networks [13], and their variants such as GRUs, are commonly applied to sequences of per-frame features, including landmark sequences. While outside the scope of the present static-alphabet system, they define the natural next step for the pipeline proposed here.

2.6 Research Gaps

A review of prior work reveals the following gaps:

- Sensor-based systems are accurate but intrusive and costly.
- Pixel-based CNN systems are sensitive to the environment and need substantial compute.
- Few lightweight, CPU-only pipelines are evaluated with per-class metrics specifically on the 26-letter ISL alphabet.

There is therefore a need for a system that:

1. Recognises ISL alphabet signs from a standard webcam without special hardware,
2. Is invariant to hand position, hand size, lighting and background, and
3. Executes in real time on CPU-only hardware with a transparent, reproducible evaluation.

The proposed ISL-HandNet addresses this gap with a normalized landmark representation and an ensemble Random Forest classifier, together with a complete training, testing and live-demo workflow.

3. SYSTEM ARCHITECTURE AND METHODOLOGY

The proposed system integrates data acquisition, spatial feature engineering and ensemble classification into a single pipeline. This section describes the architecture, the dataset, the landmark extraction and normalization steps, the classifier, the evaluation protocol and the real-time inference module.

3.1 Existing System and Its Limitations

Conventional vision-based sign recognisers pass raw RGB frames through deep networks such as MobileNetV2 [12] or ResNet [11]. Their main limitations are:

- **High computational cost:** Processing 64×64 or 224×224 RGB tensors for every frame calls for dedicated GPU infrastructure.
- **Environmental vulnerability:** Models trained on raw pixels are sensitive to background clutter, lighting changes and skin-tone variation.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151156

3.2 Proposed System Architecture (ISL-HandNet)

ISL-HandNet is built around MediaPipe landmark extraction followed by an ensemble Random Forest. A webcam frame is passed to MediaPipe Hands, which returns 21 3D joints. These are normalized relative to the wrist and palm size, flattened into a 63-element vector, and classified by the trained forest. The predicted letter is drawn on the live video. Fig 3.1 shows the runtime architecture.

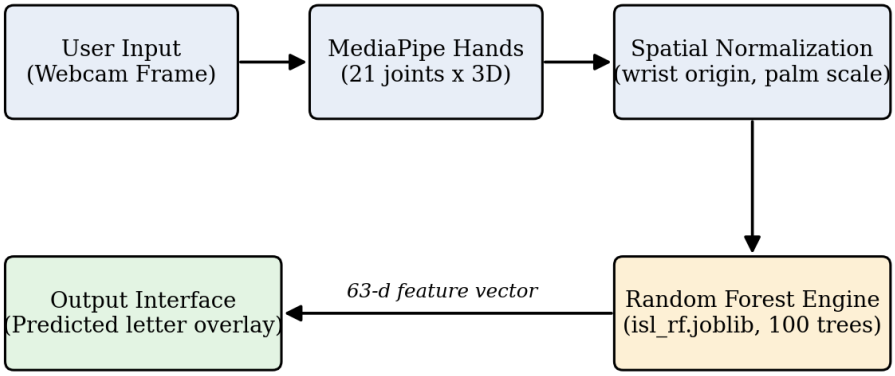


Fig 3.1: Runtime architecture of the ISL-HandNet recognition pipeline

3.3 Dataset and Data Acquisition

The project uses an ISL image dataset of 38,612 frames organised into 26 folders, one per alphabet letter (A through Z). The folder name serves as the class label. Frames are read from disk by the preparation script, so the same code path can be reused for any new image folder. Frames in which MediaPipe cannot detect a hand are rejected; the remaining frames yield 17,589 valid landmark vectors, which are stored with their labels in a single compressed archive (data/landmarks.npz).

3.4 Spatial Landmark Extraction

Each RGB frame is passed to MediaPipe Hands, configured for a single hand with a minimum detection confidence of 0.5. The model returns 21 landmarks L0, L1, …, L20, each with normalized image coordinates (x, y) and a relative depth z. Landmark 0 is the wrist, landmarks 1–4 describe the thumb, 5–8 the index finger, 9–12 the middle finger, 13–16 the ring finger and 17–20 the little finger. Fig 3.2 shows the landmark layout and the two reference points used for normalization. Flattening the landmarks gives the raw feature vector:

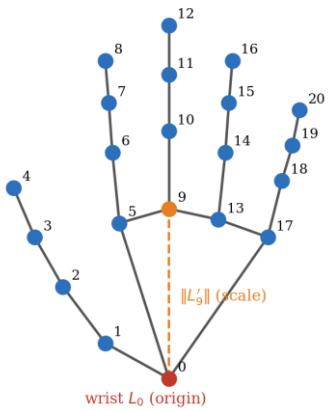


Fig 3.2: MediaPipe 21-point hand landmark layout (schematic) with the wrist origin and scale reference

$$V_raw = [x_0, y_0, z_0, x_1, y_1, z_1, \ldots, x_{20}, y_{20}, z_{20}] \quad (1)$$

3.5 Translation and Scale Normalization

Raw coordinates depend on where the hand appears in the image and how far it is from the camera, neither of which carries information about the sign. Two steps remove these effects.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151156

Translation invariance. The wrist landmark L0 is taken as the origin, and all landmarks are shifted relative to it:

$$L_i' = (x_i - x_0, y_i - y_0, z_i - z_0), \quad i = 0, 1, \ldots, 20 \qquad (2)$$

Scale invariance. The translated vector is divided by the Euclidean distance between the wrist and the middle-finger metacarpophalangeal (MCP) joint L9, which approximates the palm span:

$$V_norm = V_translated / \max(\|L_9'\|_2, \varepsilon) \qquad (3)$$

where ε is a small positive constant that prevents division by zero. In the implementation the division is performed only when the palm distance is greater than zero. After normalization, the wrist coordinates are always (0, 0, 0) and the L9 distance is always 1, so the 63-element vector captures only the relative geometry of the fingers.

3.6 Random Forest Classifier

The normalized 63-element vector is classified by a Random Forest [2] of 100 decision trees (n_estimators = 100, random_state = 42), implemented with scikit-learn [4]. Each tree is trained on a bootstrap sample of the training data and considers a random subset of features at every split. At prediction time every tree votes for a class, and the class probabilities are obtained from the fraction of votes, giving a posterior distribution over the 26 ISL alphabet classes. The following properties motivated this choice:

- **Low variance:** Averaging many decorrelated trees reduces over-fitting relative to a single decision tree.
- **Speed:** Prediction involves only simple threshold comparisons, so latency on a CPU is sub-millisecond per sample.
- **Little tuning and no feature scaling:** The landmark features are already normalized, and the default settings perform well.
- **Multi-class support:** The forest handles all 26 classes natively without one-vs-rest schemes.

3.7 Training and Evaluation Protocol

The valid landmark vectors are split into training and held-out test partitions using a stratified split (test_size = 0.2, random_state = 42), so that the class proportions are preserved in both partitions. The forest is fitted on the training partition only. Accuracy, per-class precision, recall and F1-score, and macro and weighted averages are then computed on the held-out partition, which the model never sees during training. The trained model is serialised to models/isl_rf.joblib, and the label mapping is stored in models/rf_labels.json. The automated script test_pipeline.py re-runs this evaluation on the held-out data and reports the same metrics. Fig 3.3 summarises the full workflow.

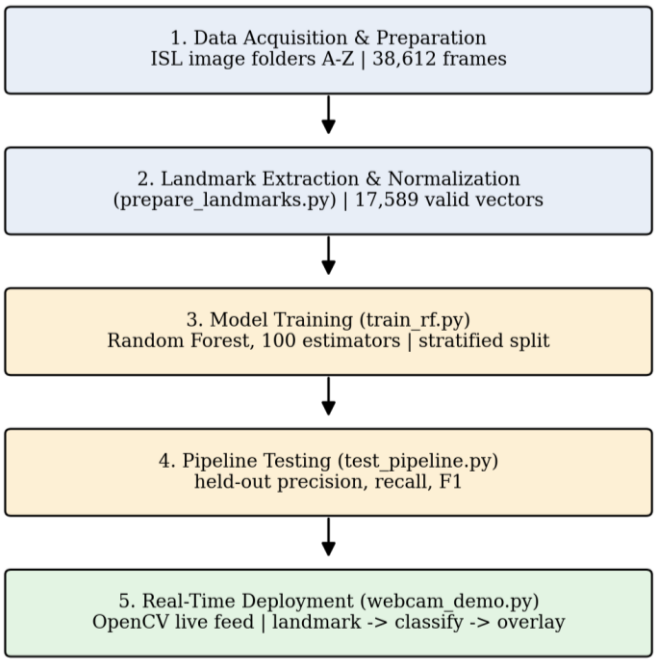


Fig 3.3: Flowchart of landmark extraction, training, testing and deployment methodology

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151156

3.8 Real-Time Inference Module

The deployment script webcam_demo.py opens the default webcam through OpenCV [5] and processes frames in a loop. For each frame, MediaPipe Hands runs in video (non-static) mode, the detected landmarks are normalized with the same procedure used in training, and the 63-element vector is passed to the saved forest. The predicted letter is rendered on the frame together with the drawn hand skeleton, and the loop ends when the user presses the q key. Using the identical normalization in training and deployment is essential, because any mismatch would shift the feature distribution and reduce live accuracy.

4. SOURCE CODE

This section outlines the core Python implementation files and architectural logic of the real-time ISL recognition pipeline, which uses MediaPipe for keypoint extraction, scikit-learn for Random Forest classification and OpenCV for real-time webcam inference.

4.1 Data Preparation and Landmark Extraction (prepare_landmarks.py)

The preprocessing module iterates through the dataset directories, extracts 21 3D joint keypoints with MediaPipe Hands, and applies wrist-relative translation and scale normalization. The driver loop that walks the A–Z folders and saves the collected arrays is omitted from the listing for brevity.

Statement / Function	Purpose in Code
mp.solutions.hands	Initializes the MediaPipe Hands detection module in static image mode.
cv2.imread() / cvtColor()	Loads raw image frames from dataset subfolders (A through Z) and converts them from BGR to RGB.
hands.process()	Detects hand landmarks and extracts 3D coordinate arrays [x, y, z].
Wrist Normalization	Shifts coordinates relative to the wrist joint (L0) for translation invariance.
Palm Scale Scaling	Divides by the middle-finger MCP distance (L9) to achieve scale invariance.
np.savez()	Compiles and saves valid landmark vectors into data/landmarks.npz.

Table 4.1: Landmark Extraction & Normalization Parameters

```
import os
import cv2
import mediapipe as mp
import numpy as np

mp_hands = mp.solutions.hands
hands = mp_hands.Hands(static_image_mode=True, max_num_hands=1, min_detection_confidence=0.5)

def extract_landmarks_from_image(image_path):
    img = cv2.imread(image_path)
    if img is None:
        return None
    image_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    result = hands.process(image_rgb)
    if not result.multi_hand_landmarks:
        return None

    hand_landmarks = result.multi_hand_landmarks[0]
    coords = np.array([[lm.x, lm.y, lm.z] for lm in hand_landmarks.landmark])

    # Normalization relative to wrist (landmark 0) and scaling
    wrist = coords[0]
    normalized = coords - wrist
```

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151156

```
    scale = np.linalg.norm(normalized[9]) # Middle finger MCP
    if scale > 0:
        normalized /= scale
    return normalized.flatten()
```

Fig 4.1: Landmark Extraction and Normalization Source Code

4.2 Model Training Module (train_rf.py)

This module configures and trains the ensemble Random Forest classifier on the extracted landmark features and reports held-out accuracy and the per-class classification report.

Statement / Function	Purpose in Code
np.load()	Loads normalized feature vectors (X) and target labels (y) from landmarks.npz.
train_test_split()	Partitions data into stratified training and held-out test sets (test_size = 0.2).
RandomForestClassifier	Initializes an ensemble model with 100 estimators for robust multi-class prediction.
accuracy_score()	Computes overall model classification accuracy on the test set.
joblib.dump()	Serializes and saves the trained classifier to models/isl_rf.joblib.

Table 4.2: Model Training Configuration Parameters

```
import numpy as np
import json
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
import joblib

def train_model():
    data = np.load('data/landmarks.npz')
    X, y = data['X'], data['y']

    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42, stratify=y
    )

    clf = RandomForestClassifier(n_estimators=100, random_state=42)
    clf.fit(X_train, y_train)

    y_pred = clf.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    print(f"Held-out test accuracy: {acc * 100:.2f}%")
    print(classification_report(y_test, y_pred))

    joblib.dump(clf, 'models/isl_rf.joblib')

if __name__ == '__main__':
    train_model()
```

Fig 4.2: Random Forest Model Training Source Code

4.3 Real-Time Webcam Inference Module (webcam_demo.py)

This script manages live video capture, real-time landmark feature normalization and instantaneous sign prediction via the trained Random Forest model.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151156

Statement / Function	Purpose in Code
cv2.VideoCapture(0)	Captures live video frames from the connected webcam feed.
hands.process()	Extracts 3D hand keypoints dynamically from live RGB frames.
clf.predict()	Passes the normalized 63-element vector into the saved Random Forest model.
cv2.putText()	Overlays the predicted alphabet sign onto the live video stream.

Table 4.3: Real-Time Deployment Configuration Parameters

```
import cv2
import joblib
import numpy as np
import mediapipe as mp

clf = joblib.load('models/isl_rf.joblib')
mp_hands = mp.solutions.hands
hands = mp_hands.Hands(static_image_mode=False, max_num_hands=1)

cap = cv2.VideoCapture(0)
while cap.isOpened():
    success, frame = cap.read()
    if not success:
        break
    image_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    results = hands.process(image_rgb)

    if results.multi_hand_landmarks:
        for hand_landmarks in results.multi_hand_landmarks:
            coords = np.array([[lm.x, lm.y, lm.z] for lm in hand_landmarks.landmark])
            normalized = coords - coords[0]
            scale = np.linalg.norm(normalized[9])
            if scale > 0:
                normalized /= scale
            features = normalized.flatten().reshape(1, -1)
            prediction = clf.predict(features)[0]

            cv2.putText(frame, f"Sign: {prediction}", (50, 50),
                        cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
            mp.solutions.drawing_utils.draw_landmarks(frame, hand_landmarks, mp_hands.HAND_CONNECTIONS)

    cv2.imshow('ISL Real-Time Recognition', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

Fig 4.3: Real-Time Webcam Inference Demo Source Code

5. EXPERIMENTAL SETUP AND RESULTS

This section describes the experimental environment and presents the evaluation of the trained Random Forest classifier on unseen test data.

5.1 Hardware and Software Setup

1. **Computing hardware:** A standard CPU-based laptop with an integrated webcam. No GPU is used for training or inference.
2. **MediaPipe Hands:** Performs hand detection and 21-point 3D landmark extraction in both static-image mode (dataset preparation) and video mode (live demo).



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151156

3. **OpenCV:** Handles image loading, colour conversion, webcam capture and on-screen overlay.
4. **NumPy and scikit-learn:** NumPy stores the feature arrays, and scikit-learn provides the Random Forest implementation, the stratified split and the evaluation metrics.
5. **joblib:** Serialises the trained classifier for use by the testing and demo scripts.

5.2 Dataset and Experimental Configuration

Table 5.1 summarises the dataset statistics and the training configuration. Of the 38,612 raw frames, 17,589 (about 45.6%) yielded a valid hand detection and were retained as landmark vectors.

Parameter	Value
Number of classes	26 (ISL alphabets A–Z)
Raw image frames	38,612
Valid landmark vectors	17,589
Feature dimension	63 (21 landmarks × x, y, z)
Normalization	Wrist origin; palm-span (L9) scaling
Classifier	Random Forest, 100 estimators
Split strategy	Stratified, test_size = 0.2, random_state = 42
Held-out test frames	3,226
Model file	models/isl_rf.joblib

Table 5.1: Dataset Statistics and Experimental Configuration

5.3 Testing Methodology

- Verify that the webcam and the Python environment are correctly set up and that the dependencies import without error.
- Run prepare_landmarks.py and confirm that the number of saved vectors and the label distribution match expectations.
- Run train_rf.py and test_pipeline.py, and check the overall accuracy and the per-class precision, recall and F1-score on the held-out set.
- Launch webcam_demo.py under normal room lighting and verify that the overlaid prediction follows the signed letter.
- If a sign is misclassified, check that the full hand is visible and well lit, and compare its class metrics in the evaluation report.

5.4 Overall Performance

Performance is measured with accuracy, precision, recall and F1-score:

$$Accuracy = (Total\ Correct\ Predictions / Total\ Samples) \times 100 \quad (4)$$

$$F1 = 2 \cdot Precision \cdot Recall / (Precision + Recall) \quad (5)$$

The Random Forest trained on 17,589 landmark vectors achieves an overall held-out accuracy of **87.10%** on 3,226 test frames. The macro-averaged precision, recall and F1-score are 0.88, 0.85 and 0.86 respectively, and the weighted averages are 0.88, 0.87 and 0.87. The classifier executes in sub-millisecond time per sample on a standard CPU, so the end-to-end real-time rate is limited by camera capture and MediaPipe landmark extraction rather than by the forest.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151156

5.5 Per-Class Classification Metrics

Table 5.2 reports precision, recall, F1-score and support (number of test frames) for each of the 26 alphabet classes.

Class	Precision	Recall	F1-Score	Support
A	0.86	0.98	0.91	137
B	0.81	0.84	0.82	64
C	0.84	0.86	0.85	59
D	0.94	0.81	0.87	115
E	0.89	0.81	0.85	73
F	0.72	0.92	0.80	123
G	0.87	0.93	0.90	135
H	0.92	1.00	0.96	150
I	0.91	0.61	0.73	162
J	0.98	0.94	0.96	114
K	0.95	0.98	0.96	161
L	0.98	0.97	0.97	186
M	0.82	0.91	0.86	182
N	0.73	0.80	0.76	176
O	0.96	0.84	0.90	76
P	0.91	0.88	0.90	146
Q	0.85	0.43	0.57	65
R	0.91	0.97	0.94	182
S	0.82	0.96	0.88	91
T	0.91	0.95	0.93	190
U	0.85	0.95	0.90	92
V	1.00	0.71	0.83	158
W	1.00	0.58	0.73	45
X	0.85	0.74	0.79	136
Y	0.73	0.96	0.83	162
Z	0.88	0.80	0.84	46
Accuracy	—	—	**0.87**	**3226**
Macro Avg	**0.88**	**0.85**	**0.86**	**3226**
Weighted Avg	**0.88**	**0.87**	**0.87**	**3226**

Table 5.2: Performance Metrics of the Random Forest Model on Unseen Test Data

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151156

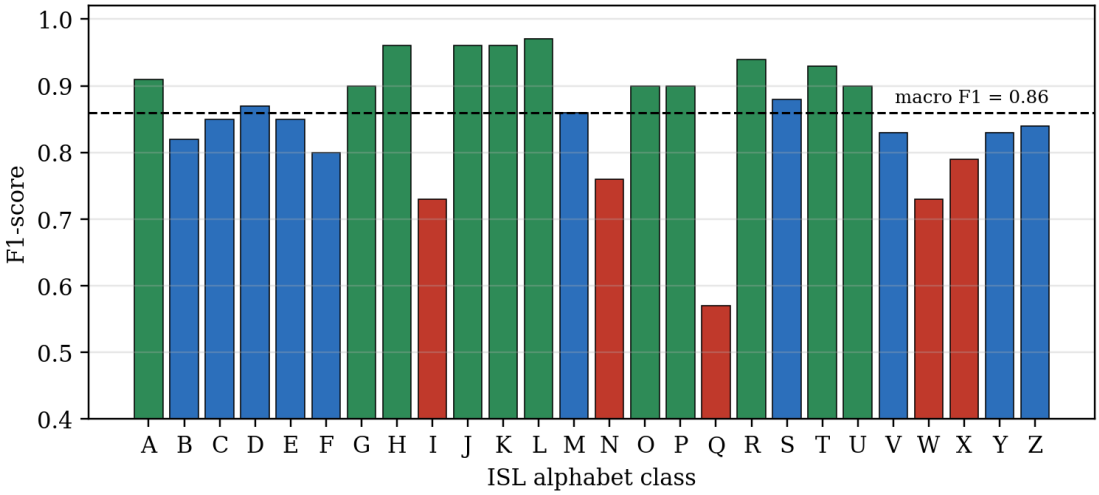


Fig 5.1: Per-class F1-score (green ≥ 0.90, blue 0.80–0.89, red < 0.80)

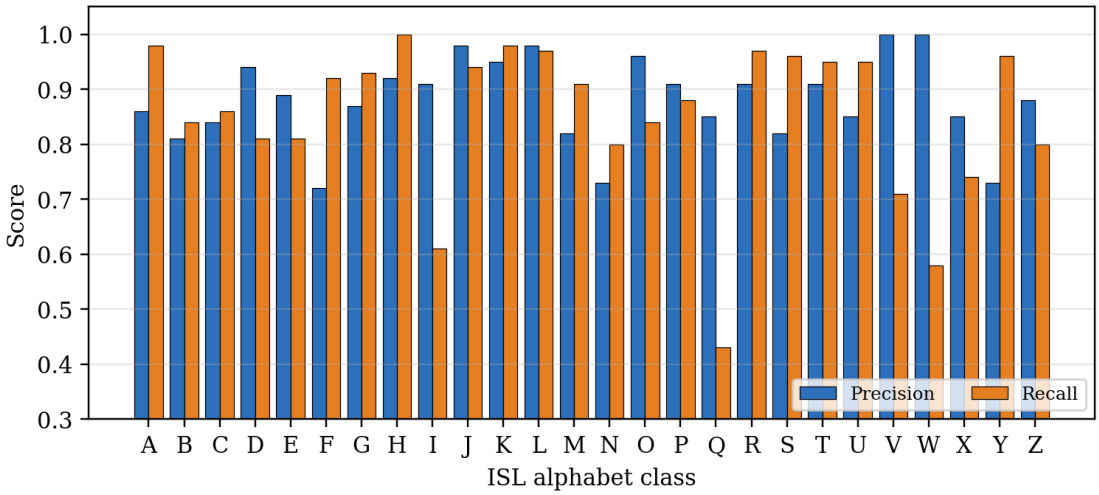


Fig 5.2: Per-class precision and recall distribution

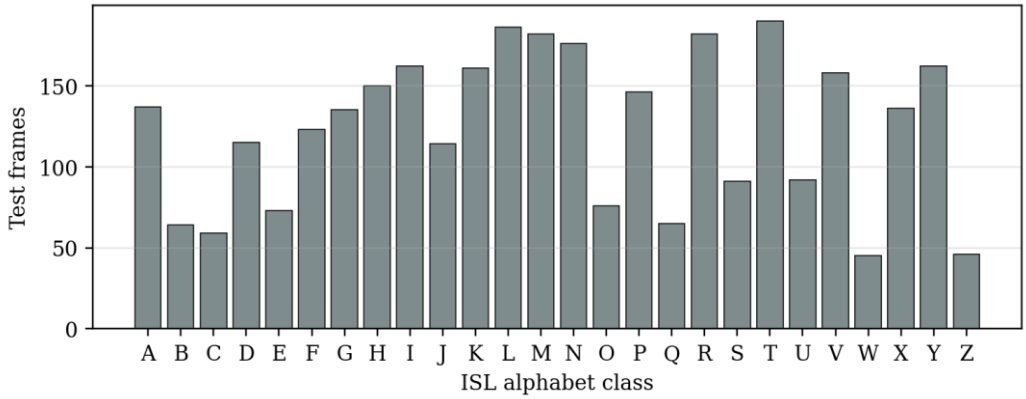


Fig 5.3: Number of held-out test frames per class

5.6 Analysis of Results

Best-performing classes. L (F1 = 0.97), H, J and K (0.96 each), R (0.94) and T (0.93) are recognised most reliably. J, K and L combine very high precision (0.95–0.98) with high recall, and H attains perfect recall (1.00). These signs have distinctive finger configurations that are well separated in the normalized landmark space.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151156

High precision, lower recall. V and W achieve perfect precision (1.00), meaning that whenever the model predicts these letters it is correct, but their recall is 0.71 and 0.58, so a considerable share of true V and W samples are assigned to other classes. W also has the smallest support (45 frames), which may have contributed.
Weakest classes. Q has the lowest scores (precision 0.85, recall 0.43, F1 = 0.57), followed by I (recall 0.61, F1 = 0.73), W (F1 = 0.73) and N (F1 = 0.76). Conversely, F, N and Y have the lowest precision (0.72–0.73), which indicates that they absorb samples that belong to other classes. The pattern of low-recall classes and low-precision classes suggests systematic confusion between handshapes with similar finger arrangements; a confusion matrix would identify the specific pairs.
Class balance. The test support ranges from 45 (W) to 190 (T) frames, so the classes are moderately imbalanced. The weighted averages (0.87 F1) are therefore slightly higher than the macro averages (0.86), because the larger classes tend to score well.

5.7 Real-Time Performance

In the live demonstration, the pipeline runs on a standard CPU and overlays the predicted letter on the webcam feed together with the hand skeleton. Since the features are derived from normalized landmarks rather than pixels, the predictions are stable across changes in background and room lighting, and they do not depend on how close the hand is to the camera, provided the hand is detected by MediaPipe.

6. DISCUSSION

This section summarises the key findings, compares the approach with pixel-based recognition, discusses the limitations observed and outlines future research directions.

6.1 Key Findings

1. **Landmark features are effective:** A 63-dimensional normalized skeleton vector is sufficient to reach 87.10% accuracy over 26 classes with a simple ensemble model, without any image-level feature learning.
2. **Invariance by construction:** Wrist-relative translation and palm-span scaling remove position and size effects before classification, so the classifier does not need to learn them from data.
3. **CPU-only real-time operation:** Both training and inference run on commodity hardware, which lowers the barrier to deploying assistive tools.
4. **Reproducible workflow:** Separate scripts for preparation, training, testing and live demonstration, with fixed random seeds, make the results easy to reproduce and extend.

6.2 Comparison with Pixel-Based Approaches

Pixel-based CNNs can reach high accuracy, but only with large, diverse image sets and GPU training, and they remain exposed to background and illumination effects. ISL-HandNet trades some of that representational capacity for robustness, speed and simplicity: the input is a 63-element vector instead of an image tensor, training takes little time, and nothing about the background or the signer's appearance reaches the classifier. A fair numerical comparison would require training a CNN on the same frames and test split, which is left for future work.

6.3 Limitations

1. **Dependence on hand detection:** Only 17,589 of 38,612 frames (about 45.6%) produced a valid landmark set. Frames with blur, occlusion or unusual orientations are lost, and at run time a failed detection yields no prediction.
2. **Single-hand tracking:** The pipeline tracks one hand (max_num_hands = 1). Signs whose meaning depends on both hands lose information, which may contribute to confusion between similar classes.
3. **Static alphabet only:** The system recognises isolated static finger-spelled letters; it does not model motion, so dynamic signs, words and sentences are not supported.
4. **Per-frame decisions:** Each frame is classified independently with no temporal smoothing, so a momentary landmark error can briefly change the displayed letter.
5. **Weak classes:** Q, I, W and N have F1-scores below 0.80, and signer diversity in the dataset is not characterised, so generalisation to unseen signers is not yet quantified.

6.4 Future Work

1. **Dynamic gesture recognition:** Extend the pipeline from static finger-spelling to continuous signs using recurrent networks such as LSTM or GRU, or Temporal Convolutional Networks, operating on sequences of landmark vectors.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151156

2. **Two-handed gesture support:** Add bimanual tracking and 126-dimensional features (two hands × 21 landmarks × 3) to support signs that require both hands.
3. **Temporal smoothing and confidence thresholds:** Require a prediction to be confident and consistent across several consecutive frames before it is accepted, reducing flicker.
4. **Richer features and models:** Add inter-joint distances and finger-angle features, and compare gradient-boosted trees and small neural networks against the Random Forest on the same split.
5. **Edge deployment:** Optimise the model for mobile and embedded devices such as Raspberry Pi or Android smartphones to provide a portable assistive tool.

7. CONCLUSION

This paper presented ISL-HandNet, a real-time Indian Sign Language alphabet recognition system that replaces heavy pixel-based CNNs with normalized MediaPipe hand landmarks and an ensemble Random Forest classifier. By expressing 21 3D keypoints relative to the wrist and scaling by palm span, the pipeline obtains translation- and scale-invariant 63-dimensional features that are independent of background and lighting. Trained on 17,589 landmark vectors, the model achieved 87.10% accuracy and a macro F1-score of 0.86 on 3,226 unseen frames across all 26 classes, with sub-millisecond classifier inference on a standard CPU. The accompanying test script and OpenCV webcam application demonstrate a complete, reproducible workflow from raw images to live prediction. The system is a low-cost foundation for assistive communication tools, and extending it with two-hand tracking, temporal models and edge deployment will move it toward recognising full ISL words and sentences.

REFERENCES

[1] C. Lugaresi et al., “MediaPipe: A framework for building perception pipelines,” arXiv preprint arXiv:1906.08172, 2019.
[2] L. Breiman, “Random forests,” Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.
[3] F. Zhang et al., “MediaPipe Hands: On-device real-time hand tracking,” in CVPR Workshop on Computer Vision for Augmented and Virtual Reality, 2020 (arXiv:2006.10214).
[4] F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.
[5] G. Bradski, “The OpenCV Library,” Dr. Dobb’s Journal of Software Tools, 2000.
[6] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in Proc. IEEE CVPR, 2005, pp. 886–893.
[7] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” International Journal of Computer Vision, vol. 60, no. 2, pp. 91–110, 2004.
[8] C. Cortes and V. Vapnik, “Support-vector networks,” Machine Learning, vol. 20, no. 3, pp. 273–297, 1995.
[9] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.
[10] A. Krizhevsky, I. Sutskever and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Advances in Neural Information Processing Systems (NeurIPS), 2012.
[11] K. He, X. Zhang, S. Ren and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE CVPR, 2016, pp. 770–778.
[12] M. Sandler et al., “MobileNetV2: Inverted residuals and linear bottlenecks,” in Proc. IEEE CVPR, 2018, pp. 4510–4520.
[13] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

 This work is licensed under a Creative Commons Attribution 4.0 International License