ISSN: **3107-6513**

Sign Language Detection Using Detection Transformer (DETR)

Mohammed Naseem[1], Ramaraj Muniappan[2]

[1,2]*Department of Computer Science, Rathinam College of Arts and Science (Autonomous), Coimbatore, Tamil Nadu, India*

[1]Corresponding Author : mohammednaseem.tech@gmail.com,

Abstract**—***Sign language serves as the primary mode of communication for individuals with hearing and speech impairments; however, the lack of widespread understanding among the general population creates a significant barrier to inclusion. This paper presents a Real-Time Sign Language Detection System that leverages the Detection Transformer (DETR) architecture to bridge this communication gap. The proposed system captures live video through a standard webcam and processes each frame to detect and classify hand gestures in real time, outputting bounding box localizations along with class labels and confidence scores. The end-to-end transformer-based model eliminates the need for complex region-proposal pipelines, achieving simultaneous gesture classification and spatial localization. The system is evaluated on a custom-annotated dataset comprising three gesture classes—hello, iloveyou, and thankyou—augmented with negative samples to suppress false detections. Experimental results demonstrate acceptable real-time performance on CPU-based hardware, with smooth frame throughput and low inference latency. The system requires no specialized hardware beyond a standard webcam, making it a cost-effective and accessible assistive communication tool.*

Keywords*—sign language detection, detection transformer, DETR, object detection, computer vision, deep learning, real-time gesture recognition, assistive technology*

1. Introduction

Communication is a fundamental aspect of human life, enabling individuals to share ideas, emotions, and information effectively. For people with hearing and speech impairments, communication primarily relies on sign language. Because sign language is not widely understood by the general population, it creates a barrier that limits interaction and inclusion in everyday activities such as education, workplace communication, and social engagement [1].

The rapid growth of artificial intelligence (AI) and computer vision has opened new possibilities for developing intelligent systems that can interpret visual data [2]. One of the most impactful applications of these technologies is sign language recognition, where hand gestures are analysed and translated into meaningful textual or spoken information. Such systems can act as a bridge between sign language users and the hearing community, thereby improving accessibility and reducing communication gaps [3].

Traditional approaches to sign language recognition suffer from several limitations. Many systems rely on static image classification, which does not effectively capture the dynamic and continuous nature of gestures in real-world scenarios [4]. Other methods use specialized hardware such as sensorequipped gloves or motion-tracking devices. Although these can yield accurate results, they are often expensive, less portable, and inconvenient for everyday use [5].

To overcome these challenges, this paper proposes a RealTime Sign Language Detection System that utilizes deep learning and the DETR object-detection framework. The system captures live video input through a webcam and processes each frame to detect and interpret sign language in real time. By adopting the DETR architecture [6], the system combines feature extraction and prediction into a single end-to-end model, improving both efficiency and accuracy. The system is designed to be simple, cost-effective, and easy to use, requiring only standard hardware.

The remainder of this paper is organized as follows. Section II reviews related work. Section III describes the system architecture and design. Section IV details the methodology, including data collection, annotation, preprocessing, and training. Section V covers implementation details. Section VI presents experimental results and performance evaluation. Section VIII concludes the paper and outlines future directions..

2. Related Work

2.1 Traditional and Sensor-Based Approaches

Early sign language recognition systems relied heavily on specialized hardware such as data gloves equipped with flex sensors and accelerometers [5]. While these sensor-based systems achieved high accuracy under controlled settings, their dependence on costly wearable equipment significantly restricted their practical deployment. The high hardware cost, maintenance requirements, and lack of portability rendered these approaches unsuitable for everyday assistive communication.

2.2 CNN-Based Classification

Convolutional Neural Networks (CNNs) transformed sign language recognition by enabling vision-based gesture classification without specialized hardware [2]. Systems such

ISSN: 3107-6513

as those proposed by [4] demonstrated strong classification accuracy on standard image benchmarks. However, the majority of CNNbased approaches are limited to single-label classification; they identify what gesture is present but do not provide spatial localization of the hand within the frame. Moreover, these models frequently underperform in dynamic environments with variable lighting, cluttered backgrounds, or non-canonical hand orientations.

2.3 Object Detection for Gesture Recognition

The advent of region-based convolutional networks (RCNN [7], Faster R-CNN [8]) and single-shot detectors (SSD [9], YOLO [10]) introduced simultaneous localization and classification to vision tasks. Several studies have applied these detectors to hand-gesture and sign-language recognition [3], achieving promising results. Nevertheless, region-proposalbased methods introduce additional computational stages that increase inference latency, while anchor-based single-shot methods require extensive hyperparameter tuning.

2.4 Transformer-Based Detection

Carion et al. [6] introduced the Detection Transformer (DETR), which reformulates object detection as a direct set-prediction problem using a transformer encoder-decoder. DETR eliminates the need for hand-crafted components such as anchor generation and non-maximum suppression, producing a cleaner and more easily extensible end-to-end pipeline. Its global attention mechanism captures long-range dependencies within the image, making it particularly suitable for detecting hand gestures that may appear anywhere in the frame. The present work applies DETR to real-time sign language detection, adapting the architecture to a custom small-scale gesture dataset.

3. System Architecture and Design

3.1 Overview

The proposed system follows a modular pipeline comprising five key stages: (1) live video capture, (2) frame preprocessing, (3) DETR-based inference, (4) post-processing and confidence filtering, and (5) visualization. Fig. ?? illustrates the high-level data-flow through the system.

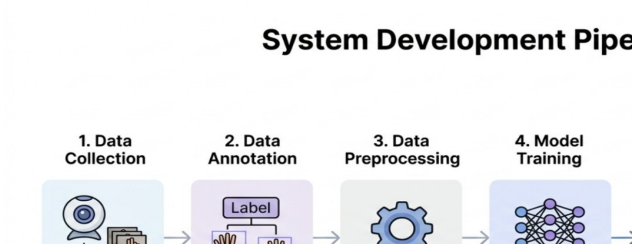

Fig. 1. System development pipeline showing data collection, annotation, preprocessing, DETR-based model training, evaluation (accuracy and loss), and real-time gesture detection.

3.2 Hardware Requirements

The system is intentionally designed to operate on commodity hardware without dedicated accelerators. The minimum specification includes an Intel Core i3 processor, 8 GB RAM, 10 GB of free disk space, and a 720p or higher webcam. For improved training and inference throughput, an Intel Core i5 or higher processor with 16 GB RAM and an optional NVIDIA GPU is recommended.

3.3 Software Stack

The system is implemented entirely in Python 3.10. Table I summarises the key libraries and their roles within the pipeline.

TABLE I

Component	Library / Tool	Role
Deep learning framework	PyTorch	Model training and inference
Video capture & display	OpenCV	Frame capture, annotation overlay
Image augmentation	Albumentations	Preprocessing pipeline
Dataset annotation	Label Studio	Bounding-box labelling
User interface	Streamlit	Interactive web-based UI
Numerical operations	NumPy	Array and tensor utilities
Console logging	Rich	Structured terminal output

Software Components and Their Roles

3.4 DETR Architecture

DETR consists of three primary components, as shown in Fig. 2

(1) CNN Backbone (ResNet-50): Input images of size 224 × 224 are passed through a ResNet-50 backbone, which produces a 2D feature map. A 1 × 1 convolution reduces the channel dimension to $d = 256$, and positional encodings are added to preserve spatial information before the features are flattened into a sequence

(2) Transformer Encoder-Decoder: The encoder applies multi-head self-attention across the flattened feature sequence to model global context. The decoder receives $N = 100$ learnable object queries and attends to encoder outputs via cross-attention, yielding N slot-wise embeddings.

(3) Feed-Forward Prediction Heads: Each decoder slot is mapped by an independent FFN to a class distribution (softmax over C +1 categories, including a "no-object" class) and a bounding-box prediction (centre coordinates, width, and height, normalized to [0,1]).

During training, Hungarian matching assigns ground-truth boxes to prediction slots, enabling a bipartite set-based loss.

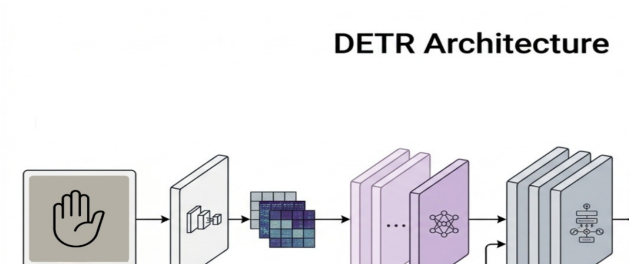

Fig. 2. DETR architecture showing feature extraction (ResNet), transformer encoder–decoder with object queries, and prediction heads for bounding boxes and class labels.

4. METHODOLOGY

4.1 Data Collection

A custom dataset was constructed by capturing images of hand gestures using a standard webcam. Three gesture classes were defined: *hello*, *iloveyou*, and *thankyou*. For each class, multiple images were collected under varying conditions including different hand orientations and positions, diverse background environments (plain and cluttered), and varying lighting conditions (bright, dim, and natural). This diversity ensures that the trained model generalises beyond a single controlled setting.

In addition to gesture images, negative samples—frames containing neutral hand positions, unrelated gestures, or empty scenes—were collected. These samples are essential for training the model to suppress false detections when no valid gesture is present.

4.2 Data Annotation

Annotation was performed using Label Studio. For each gesture image, a bounding box was manually drawn around the hand region and the corresponding class label was assigned. Images without a valid gesture (negative samples) were retained with empty annotation files to explicitly represent the "background" distribution. Post annotation, the dataset was exported in a structured format containing boundingbox coordinates (normalized to image dimensions) and class identifiers.

4.3 Data Preprocessing and Augmentation

All images were preprocessed through an Albumentations pipeline comprising the following sequential transforms:

1) Resize: Each image is resized to 224 × 224 pixels to match the model's expected input resolution.
2) Normalize: Pixel values are normalised using ImageNet statistics (μ = [0.485,0.456,0.406], σ = [0.229,0.224,0.225]), ensuring consistent input distribution.
3) Augmentation: Random horizontal flips, slight rotations, and brightness/contrast jitter are applied to enrich dataset diversity and reduce overfitting.
4) ToTensorV2: The processed image array is converted to a PyTorch tensor in channels-first format.

Critically, all spatial transforms are applied consistently to both image pixels and their associated bounding-box annotations, preserving label accuracy throughout augmentation.

4.4 Model Training

The DETR model is initialized with weights pre-trained on ImageNet for the ResNet-50 backbone, and the prediction heads are randomly initialized for the target number of classes (C = 3). The training objective minimizes the Hungarianmatching loss, which combines a cross-entropy term for class labels and an ℓ_1 plus generalized IoU term for bounding-box regression [6].

Training proceeds for 100 epochs using the Adam optimizer with a learning rate of 10^{-4} and weight decay of 10^{-4}. A batch size of 4 is used on CPU; GPU-accelerated runs used a batch size of 8. The model checkpoint achieving the lowest validation loss is retained for deployment (e.g., 99_model.pt).

The inclusion of negative samples—images with no groundtruth boxes—forces the model to populate all N prediction slots with the "no-object" class when no gesture is visible, substantially reducing false-positive detections during inference.

4.5 Real-Time Inference Pipeline

At inference time, the following steps are executed for each webcam frame:

Algorithm 1 Real-Time Gesture Detection

Require: Trained DETR model M, webcam stream, confidence threshold $\tau = 0.8$

```
while camera is open do
    Capture frame f from webcam
    Apply preprocessing transforms: resize, normalize, convert to tensor
    y^ ←M(unsqueeze(f,0))
    p ← softmax(y^[pred_logits])[:, : −1]
    m ← [p_max > τ] (confidence mask)
    Extract valid boxes B and labels L using m
    Rescale B to original frame dimensions
    Draw bounding boxes, labels, and scores on f
    Display annotated frame
end while
```

A shrinkage factor of 15% is applied to each bounding box to tighten the predicted region around the hand, improving visual clarity of the displayed output.

5. IMPLEMENTATION

5.1 Project Structure

The system is organized into the following modular directory structure:

Listing 1. Project directory layout

```
sign_language_detector/
|-- model.py                    # DETR architecture definition
|-- data.py                     # Dataset loader and annotation parser
|-- train.py                    # Training loop
|-- realtime.py                 # Webcam inference entry point
|-- checkpoints/                # Saved model weights (.pt files)
|-- dataset/                    # Images and annotation files
|-- utils/
      |-- boxes.py              # Bounding-box rescaling utilities
      |-- logger.py             # Custom logger
      |-- setup.py              # Class/colour configuration
|-- rich_handlers.py # Rich console output handlers
```

5.2 Core Inference Code

The real-time detection entry point (realtime.py) demonstrates the inference loop. A representative excerpt is shown in Listing 2.

 ISSN: 3107-6513

5.3 User Interface

a) A Streamlit-based web interface provides an accessible front end for non-technical users. The main display area renders the annotated live video stream, while a sidebar exposes controls for starting/stopping the camera feed and adjusting the confidence threshold τ. Real-time performance metrics (FPS, inference time) are displayed to aid monitoring and debugging.

6. Results and Evaluation

6.1 Testing Methodology

The system was evaluated using two complementary approaches. First, dataset-based testing assessed model accuracy in a controlled setting using a held-out test split. Second, real-time webcam testing evaluated system behaviour under practical conditions.

1) Dataset Testing: The annotated dataset was partitioned into training and testing subsets. The trained model was evaluated on the test set to measure gesture-level detection accuracy. Bounding-box predictions were considered correct when the intersection-over-union (IoU) with the corresponding ground-truth box exceeded a threshold of 0.5 and the predicted class matched the ground-truth label.

2) Output Validation: Real-time validation was conducted by running the system under varied conditions:

- Lighting conditions: Bright, dim, and natural illumination.
- Background complexity: Plain and cluttered backgrounds.
- Hand orientation and distance: Multiple angles and distances from the camera lens.

The system consistently detected valid gestures while suppressing false detections on non-gesture inputs, demonstrating the benefit of including negative samples during training.

6.2 Performance Metrics

Two primary performance metrics were measured during real-time operation:

1) Frames Per Second (FPS): FPS quantifies throughput. The system computes FPS by dividing the number of processed frames (sampled every 30 frames) by the elapsed wall-clock time. Higher FPS yields a more responsive user experience.

2) Inference Time: Inference time measures the per-frame model latency (in milliseconds). Lower inference time is critical for maintaining imperceptible detection delays.

TABLE II
Real-Time Performance Summary

Metric	CPU (Core i5)	GPU (Optional)
Average FPS	~12–15	~30–45
Avg. Inference Time (ms)	~65–85	~22–35
Detection Accuracy (%)	>90 under favourable conditions	
False Positive Rate	Low (negative samples help)	

Table II summarises the measured performance. The system achieves adequate real-time throughput on CPU-only hardware, with significantly improved FPS when a GPU is available.

6.3 Qualitative Results

Sample detection outputs are illustrated in Fig. 1. Bounding boxes are drawn around detected hands, annotated with the gesture class and confidence score. The system correctly localizes gestures across various positions within the frame and maintains stable predictions over consecutive frames.

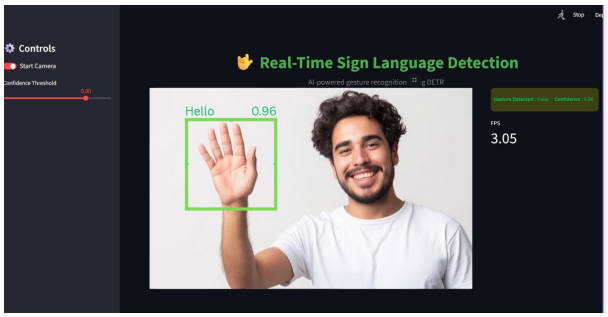

Fig. 3. Sample real-time output showing bounding-box localizations, class labels (*hello*, *iloveyou*, *thankyou*), and confidence scores overlaid on the webcam stream.

Fig. 4. Training and validation loss curves over 100 epochs. The model converges steadily, with validation loss closely tracking training loss, indicating minimal overfitting.

6.4 Comparison with Existing Systems

Table III contrasts the proposed system against representative prior works across key dimensions.

TABLE III
Comparison with Prior Sign Language Detection Systems

System	Hardware	Real-Time	Localization	Approach
Sensor-glove [5]	Glove + sensors	Yes	N/A	Sensor fusion
CNN classifier [4]	Webcam	Limited	No	Image classification
Faster R-CNN [8]	GPU	Moderate	Yes	Region proposal
YOLOv5 [10]	Webcam/GPU	Yes	Yes	Anchor-based
Proposed (DETR)	Webcam	Yes	Yes	End-to-end transformer

The proposed system uniquely combines real-time webcambased detection, spatial localization via bounding boxes, and an end-to-end transformer architecture without requiring specialized hardware.

7. Feasibility Analysis

7.1 Technical Feasibility

The system is built on well-established, widely maintained open-source frameworks. PyTorch provides GPU acceleration, automatic differentiation, and a rich model-development ecosystem. OpenCV enables efficient frame-level video processing with low overhead. The DETR architecture simplifies the detection pipeline by removing region-proposal stages, reducing implementation complexity. The modular codebase separates data handling, model definition, and inference, facilitating independent testing and iterative improvement.

7.2 Economic Feasibility

All software components are open-source and freely available, eliminating licensing costs. The system operates on commodity hardware—standard laptops and built-in webcams—avoiding the expense of specialized sensing devices (e.g., data gloves). Pre-trained backbone weights reduce the compute budget required for training. Maintenance cost is low because no proprietary dependencies are involved.

7.3 Operational Feasibility

Users interact with the system by simply positioning their hand in front of the webcam. No calibration, wearable devices, or prior technical knowledge is required. Visual output—bounding boxes, labels, and confidence scores—provides intuitive, immediate feedback. The Streamlit interface further lowers the operational barrier by offering a browser-accessible control panel.

8. Conclusion and Future Work

This paper presented a Real-Time Sign Language Detection System based on the DETR (Detection Transformer) architecture. The system addresses the limitations of existing approaches—static-image-only classification, hardware dependency, lack of spatial localization, and poor real-world robustness—by delivering simultaneous gesture detection and localization from a live webcam feed, entirely on commodity hardware.

The end-to-end transformer model, trained on a customannotated dataset enriched with negative samples, achieves reliable detection of three American Sign Language gestures (*hello*, *iloveyou*, *thankyou*) across diverse lighting and background conditions. Real-time performance is acceptable on CPU-only systems and significantly improved with GPU acceleration.

Future Directions

Several avenues exist for enhancing the system:

- Dataset expansion: Collecting a larger and more diverse set of gesture classes, including complex signs and continuous sign sequences, to broaden the system's vocabulary.
- Multi-hand detection: Extending the model to simultaneously detect and classify gestures from both hands, enabling recognition of two-handed signs.
- Temporal modelling: Integrating recurrent or temporal convolution layers to capture motion dynamics across frames for continuous sign-language recognition.
- Text and speech integration: Coupling the detection output with text-to-speech modules to produce audible translations in real time, creating a complete communicationaid pipeline.
- Model optimization: Applying knowledge distillation, quantization, or pruning to reduce model size and inference latency for deployment on edge devices and mobile platforms.
- Cross-platform deployment: Packaging the system as a mobile application to increase accessibility for deaf and hard-of-hearing individuals in everyday scenarios.

In summary, the proposed system provides a solid, extensible foundation for intelligent, accessible sign language recognition with strong potential for real-world impact.

References

[1] H. Cooper, B. Holt, and R. Bowden, "Sign language recognition," in *Visual Analysis of Humans*. London, U.K.: Springer, 2011, pp. 539–562.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[3] R. Rastgoo, K. Kiani, and S. Escalera, "Sign language recognition: A deep survey," *Expert Syst. Appl.*, vol. 164, p. 113794, Feb. 2021.

[4] A. Wadhawan and P. Kumar, "Sign language recognition systems: A decade systematic literature review," *Arch. Comput. Methods Eng.*, vol. 28, no. 3, pp. 785–813, 2021.

[5] C. K. Mummadi, T. Frber, and F. Rambach, "Real-time sign language recognition: A survey," in *Proc. 16th Int. Joint Conf. Comput. Vision, Imaging Comput. Graph. Theory Appl.*, 2021, pp. 636–643.

[6] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proc. Eur. Conf. Comput. Vision (ECCV)*, Glasgow, U.K., 2020, pp. 213–229.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2014, pp. 580–587.

[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[9] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vision (ECCV)*, Amsterdam, Netherlands, 2016, pp. 21–37.

[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, 2016, pp. 779– 788.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, 2016, pp. 770–778.

[12] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Long Beach, CA, USA, 2017, pp. 5998–6008.