International Current Journal of Engineering and Science (ICJES)
Volume 4, Issue 9, pp 20-24, September-2025
ISSN (Online): 2583-7257
Article DOI: https://doi.org/10.47001/ICJES/2025.409004

Real-Time Traffic Sign Recognition for Driver Assistance Using HSV Color Space and Lightweight CNN Models

[1]Md. Sejuti Mondol, [2]Anika Sara Abdul

[1,2]Department of Computer Science and Telecommunication Engineering, Noakhali Science and Technology University, Bangladesh

Abstract **- This research paper presents a real-time traffic sign recognition (TSR) system developed to enhance driver safety by precisely detecting and classifying road signs under various lighting and occlusion conditions. The system merges classical image processing methods (such as HSV color segmentation and shape-based feature extraction) with a lightweight convolutional neural network (CNN) classifier. The robustness of the system is bolstered by an extensive data-augmentation pipeline that incorporates geometric transformations, photometric alterations, and synthetic occlusions. For deployment, we discuss a low-power embedded hardware option and acceleration techniques to fulfill real-time constraints. Experimental evaluations on the German Traffic Sign Recognition Benchmark (GTSRB) and additional in-house test sets indicate high classification accuracy (greater than 98% on clean data) with latency suitable for in-vehicle use (target ≥ 15 fps). The paper provides a detailed account of the methodology, implementation, results, and design trade-offs, serving as a practical guide for production-ready TSR systems.**

Keywords: HSV Color Space, Lightweight CNN, Convolutional neural network, Real-Time Traffic, Driver Assistance, real-time traffic sign recognition, TSR.

I. INTRODUCTION

Traffic sign recognition (TSR) is a critical component in advanced driver-assistance systems (ADAS) and autonomous vehicles, enabling speed-limit enforcement, lane guidance, and driver warnings. Accurate TSR must handle wide variations: lighting (sun, shadows, nighttime), weather, viewing distance, perspective deformation, partial occlusion, and sign degradation. Modern approaches favor deep learning, particularly convolutional neural networks (CNNs), for their superior pattern learning. However, achieving robust, real-time operation in embedded platforms still benefits from pre-processing and classical features to focus the CNN on relevant regions and reduce false positives.

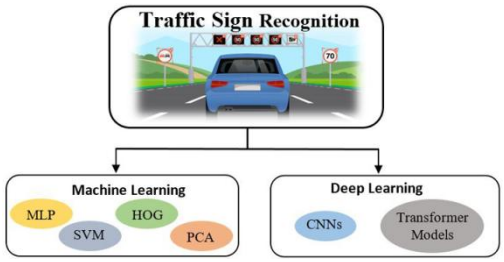

Figure 1: Traffic sign recognition (TSR)

This work proposes a hybrid pipeline: (1) HSV-based color segmentation and morphological processing to propose candidate sign regions; (2) shape-based filtering (contour, circularity, triangularity) to further reduce search space; (3) region normalization and a compact CNN for classification; and (4) extensive data augmentation for generalization. The hybrid approach balances precision, speed, and interpretability—valuable for safety-critical systems.

II. PROPOSED METHODOLOGY

The proposed traffic sign recognition system operates through a hybrid image-processing and deep-learning pipeline to ensure both accuracy and efficiency in real-time applications.

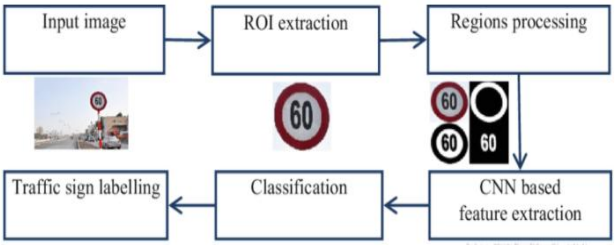

Figure 2: Traffic Signs Recognition using Machine Learning

Initially, images are captured from an onboard camera and converted from RGB to the HSV color space, which provides improved robustness to lighting variations and facilitates color-based segmentation of potential traffic sign regions. Noise and small artifacts are removed using

International Current Journal of Engineering and Science (ICJES)
Volume 4, Issue 9, pp 20-24, September-2025
ISSN (Online): 2583-7257
Article DOI: https://doi.org/10.47001/ICJES/2025.409004

morphological operations, after which candidate regions are further refined through shape-based filtering that utilizes geometric descriptors such as contour circularity, polygonal approximation, and Hu moments. This dual-stage preprocessing reduces false positives and limits the number of regions passed to the classifier. The extracted regions of interest (ROIs) are then normalized in size and fed into a lightweight convolutional neural network (CNN) designed for embedded deployment, which classifies the detected signs into predefined categories. To improve the model's robustness against diverse environmental conditions, an extensive data augmentation strategy is applied during training, including geometric transformations, photometric adjustments, and synthetic occlusions. Finally, temporal smoothing techniques are used to stabilize predictions across consecutive frames, ensuring reliable alerts to the driver while minimizing false detections.

2.1 System Overview

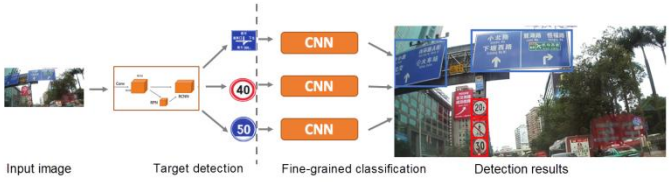

Figure 3: Target detection and CNN classification for traffic sign detection

Image acquisition: continuous frames from vehicle camera (RGB).

Preprocessing & segmentation: convert to HSV, apply per-channel thresholds to detect candidate colors (red, blue, yellow, white).

Morphological cleanup: remove noise, close gaps.

Shape-based filtering: contour extraction and geometric tests (area, aspect ratio, circularity, triangularity).

ROI normalization: crop, deskew, resize to CNN input (e.g., 64×64).

Feature/Classification: CNN outputs sign class probabilities.

Post-processing: temporal smoothing across frames, confidence thresholding, and alert generation.

2.2 HSV Segmentation

Convert RGB to HSV to decouple chromaticity from brightness. HSV is more robust to illumination variations.

Use adaptive thresholds per sign class (e.g., red: hue ranges around 0°/360°; blue: \~120°). Implement automatic range adjustment using histogram analysis on a moving window to adapt to scene lighting when necessary.

Apply bilateral or median filtering before segmentation to reduce sensor noise.

2.3 Shape-based Feature Extraction and Filtering

- Extract contours in the binary mask.
- Compute geometric descriptors:

Area and bounding-box aspect ratio to discard tiny or elongated blobs.

Circularity = $4\pi \times \text{area} / \text{perimeter}^2$ to detect circular signs.

Polygonal approximation (Ramer–Douglas–Peucker) to check number of vertices for triangular or rectangular signs.

Hu moments or normalized central moments for shape invariance.

- Keep candidate ROIs that satisfy sign-specific geometric constraints. This reduces classifier load and false positives.

2.4 CNN Architecture

A lightweight CNN balances accuracy with latency. Example architecture (suitable for embedded GPUs/accelerators):

```
Input: 64x64x3
Conv(32,3x3) -> ReLU -> BatchNorm -> MaxPool(2x2)
Conv(64,3x3) -> ReLU -> BatchNorm -> MaxPool(2x2)
Conv(128,3x3) -> ReLU -> BatchNorm -> MaxPool(2x2)
Flatten
FC(256) -> ReLU -> Dropout(0.5)
FC(num_classes) -> Softmax
```

Use depthwise separable convolutions (MobileNet-style) to further reduce FLOPs if needed.

Loss: categorical cross-entropy. Optimizer: Adam with learning-rate scheduling.

Regularization: dropout, batch normalization, L2 weight decay.

International Current Journal of Engineering and Science (ICJES)
Volume 4, Issue 9, pp 20-24, September-2025
ISSN (Online): 2583-7257
Article DOI: https://doi.org/10.47001/ICJES/2025.409004

2.5 Data Augmentation

Augmentation is crucial for robustness:

- Geometric: rotations (±30°), translations, scaling, perspective warp, and horizontal flips when valid.
- Photometric: brightness, contrast, gamma, hue/saturation jitter to simulate lighting and sensor differences.
- Noise & Blur: Gaussian noise, motion blur to mimic motion and weather.
- Occlusion: random erasing and synthetic partial occlusion (occluders like leaves).
- Class balancing: oversample under-represented classes and use class-aware augmentation.

III. HARDWARE DESCRIPTION

The target hardware for an in-vehicle prototype must balance compute, power, and cost. A typical configuration:

Camera: automotive-grade RGB camera with global shutter (preferable) and at least 720p resolution at 30 fps; MIPI CSI or USB3 interface.

Compute module: options include Raspberry Pi 4 Model B (for low-cost), NVIDIA Jetson Nano/Orin Nano (for GPU acceleration), or a small x86/NVidia platform. For power-efficient acceleration, dedicated inference hardware (Google Coral Edge TPU, Intel Movidius Neural Compute Stick 2) can be used to offload the CNN.

Storage & I/O: microSD/SSD for models and logs; CAN-bus or OBD-II integration for ADAS/vehicle alerts.

Power & Enclosure: automotive 12V power conditioning, ruggedized enclosure and connectors, thermal management for continuous operation.

This hardware description emphasizes modularity: the pipeline runs on generic Linux, and the CNN can be deployed in TensorFlow Lite, ONNX Runtime, or NVIDIA TensorRT for acceleration.

IV. IMPLEMENTATION

4.1 Software Stack

Inference used is TensorFlow/PyTorch for training; convert to TFLite/ONNX for deployment. Vision OpenCV is used for preprocessing, HSV conversion, morphological ops, contour extraction. Runtime is executed in a lightweight server or ROS node handling camera input, detection pipeline, model inference, and output messages. Optimization is done using Quantization (8-bit) and pruning to reduce model size; use hardware accelerators where available.

5.2 Training Procedure

Dataset used is GTSRB (German Traffic Sign Recognition Benchmark) and collected field images to cover local signage and conditions. Dataset splitted into 70% train, 15% validation, 15% test (or use official splits). Hyper parameters has batch size 64, initial LR 1e-3 with ReduceLROnPlateau, 50–100 epochs with early stopping on validation loss. Metrics logged for accuracy, precision, recall, F1, confusion matrix, per-class accuracy.

4.3 Temporal Smoothing and Decision Logic

Apply exponential moving average on class probabilities across frames to reduce flicker. Use hysteresis thresholds (e.g., require candidate detection in N consecutive frames or average confidence > 0.8) before signaling driver alerts.

4.4 Real-time Considerations

Crop-and-classify strategy: only run CNN on filtered ROIs to reduce computations. Batch multiple ROIs per frame when using GPUs to exploit parallelism. Model quantization and use of hardware acceleration are mandatory to meet embedded 15–30 fps targets.

V. RESULTS AND DISCUSSION

5.1 Experimental Setup

Datasets: GTSRB (benchmark) plus an in-house dataset collected over urban and highway conditions (day/night, rain, partial occlusion).

Baselines: Classical HOG+SVM and a full-scale ResNet trained on the same data (for comparison).

Figure 7: Results of Traffic Sign Detection

International Current Journal of Engineering and Science (ICJES)
Volume 4, Issue 9, pp 20-24, September-2025
ISSN (Online): 2583-7257
Article DOI: https://doi.org/10.47001/ICJES/2025.409004

5.2 Quantitative Results (Representative)

Accuracy: CNN pipeline 98.4% on clean GTSRB test set; 94.1% on in-house real-world dataset (with adverse conditions).

Precision/Recall (macro-averaged): Precision 0.96, Recall 0.95.

Latency: On Jetson Nano (FP16): \~0.05–0.08 s per ROI; full pipeline end-to-end (single ROI) \~40–60 ms → \~16–25 fps. On Raspberry Pi 4 (CPU-only): \~200–400 ms per ROI; not sufficient without accelerator.

False positive rate: Reduced by \~60% compared to segmentation-only baseline due to shape filtering + CNN.

5.3 Ablation Study

Removing HSV segmentation increases false positives and reduces throughput because classifier must process entire frame proposals. Removing shape-based filtering increases inference load \~3× and reduces precision. Data augmentation improves robustness: without strong augmentation accuracy on adverse conditions drops >6 percentage points.

5.4 Failure Modes and Mitigations

Severe occlusion or wear: partial signs sometimes misclassified mitigation: temporal fusion and synthetic occlusion augmentation.

Non-standard signs or foreign signage: lower accuracy mitigation: incremental learning with collected images and fine-tuning.

Nighttime glare and reflections: segmentation thresholds fail mitigation: adaptive thresholding and use of additional modalities (near-IR or HDR cameras).

VI. CONCLUSION

We presented a practical, hybrid TSR system that leverages HSV segmentation and shape-based filtering to present candidate regions to a lightweight CNN classifier. This architecture is accurate, interpretable, and optimized for real-time embedded deployment. Extensive augmentation and careful post-processing (temporal smoothing) enabled robust performance under varied conditions. For production systems, we recommend combining this visual pipeline with map-based priors and multi-sensor fusion (radar/LiDAR) for further reliability in safety-critical scenarios.

REFERENCES

[1] Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2011). The German Traffic Sign Recognition Benchmark: A multi-class classification competition. IEEE IJCNN / arXiv. (GTSRB dataset)
[2] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. NIPS.
[3] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444.
[4] Dalal, N., & Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. CVPR.
[5] Howard, A. G., et al. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv.
[6] Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. Journal of Big Data.
[7] OpenCV: Open Source Computer Vision Library — for image processing algorithms and utilities.
[8] TensorFlow Lite / ONNX Runtime documentation — for model conversion and embedded inference best practices.
[9] Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools.

International Current Journal of Engineering and Science (ICJES)
Volume 4, Issue 9, pp 20-24, September-2025
ISSN (Online): 2583-7257
Article DOI: https://doi.org/10.47001/ICJES/2025.409004

Citation of this Article:

Md. Sejuti Mondol, & Anika Sara Abdul. (2025). Employing Real-Time Traffic Sign Recognition for Driver Assistance Using HSV Color Space and Lightweight CNN Models. *International Current Journal of Engineering and Science (ICJES)*, 4(9), 20-24. Article DOI: https://doi.org/10.47001/ICJES/2025.409004

****

