INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 2, February 2026 | ISSN: 2456-4184 | IJNRD.ORG

Deep Learning-Based Traffic Sign Recognition System

Mrs. G. Sangeetha Lakshmi [1], V. Ezhilarasi[2]

Mrs. G. Sangeetha Lakshmi [1], M.S(IT)., M.Phil., Assistant Professor and Head,

Department of Computer Science and Applications

D.K.M. College for Women (Autonomous)

V. Ezhilarasi[2], PG Student, Department of Computer Science and Applications

Department of Computer Science and Applications

D.K.M. College for Women (Autonomous), Vellore, Tamil Nadu, India.

Abstract—This paper presents a deep learning-based Traffic Sign Recognition (TSR) system designed to support intelligent transportation systems and autonomous driving applications. The proposed system employs Convolutional Neural Networks (CNNs) trained on the German Traffic Sign Recognition Benchmark (GTSRB) dataset comprising 43 sign classes. Image preprocessing techniques including resizing, normalization, and augmentation are applied to enhance feature learning and model robustness. Multiple deep learning architectures—including VGG16, ResNet50, MobileNet, and Inception V3—are evaluated to identify the most effective model. Transfer learning is leveraged to reduce training time and improve accuracy, especially for under-represented sign classes. Performance is assessed using accuracy, precision, recall, F1-score, and confusion matrix under conditions including noise, blur, and occlusion. The proposed system achieves a classification accuracy of 97.8% on the GTSRB test set, demonstrating strong generalization capability. A Flask-based web application enables real-time image upload and prediction, making the system practically deployable. Results highlight the effectiveness of deep learning in automating traffic sign recognition for real-world road safety applications.

Keywords—*Traffic Sign Recognition, Convolutional Neural Network, Deep Learning, Transfer Learning, GTSRB, Autonomous Vehicles, ADAS, Image Classification.*

I. INTRODUCTION

Traffic signs are essential components of modern road transportation systems, providing drivers with critical information regarding speed limits, warnings, prohibitions, and directions. Proper recognition and interpretation of these signs are vital for ensuring road safety and preventing accidents. However, drivers may fail to notice or correctly interpret traffic signs due to fatigue, distraction, poor visibility, adverse weather conditions, or high-speed driving. To overcome these limitations, automated Traffic Sign Recognition (TSR) systems have been developed as a key component of intelligent transportation systems and Advanced Driver Assistance Systems (ADAS).

In recent years, deep learning has significantly transformed the field of computer vision. Traditional traffic sign recognition methods relied on handcrafted features such as color segmentation, edge detection, and shape analysis, which often struggled in complex real-world environments. In contrast, deep learning techniques—particularly Convolutional Neural Networks (CNNs)—automatically learn hierarchical feature representations directly from raw image data. This enables the system to achieve higher accuracy and robustness under varying lighting conditions, occlusions, background noise, and motion blur [1].

Deep learning-based TSR systems are typically trained using large benchmark datasets such as the German Traffic Sign Recognition Benchmark (GTSRB) and the Belgian Traffic Sign Classification Benchmark, which contain thousands of labeled traffic sign images across multiple categories. These systems generally operate in two main stages: detection, where traffic signs are located within an image or video frame, and classification, where the detected signs are categorized into predefined classes.

This paper proposes an end-to-end CNN-based system

E. Web Application Deployment

A Flask-based web application provides a user-friendly interface for real-time traffic sign recognition. The system supports JWT-based user authentication, secure image upload, inference using the saved Keras model, and scan history logging to a SQLite database via SQLAlchemy. The /predict REST endpoint accepts POST requests with image files, preprocesses the image to 50x50 pixels, normalizes pixel values, performs model inference, and returns the recognized sign label along with a confidence score as JSON. User scan history is stored and accessible via a dedicated /history endpoint for performance monitoring.

V. SYSTEM DESIGN

A. System Architecture

The overall system architecture consists of five interconnected components: (1) Input Interface—a camera or web-based image upload module; (2) Preprocessing Module—resizing, normalization, and optional augmentation; (3) Deep Learning Engine—the trained CNN or VGG16 model for feature extraction and classification; (4) Output Module—displays the recognized sign label and confidence score to the user; and (5) Logging Module—records prediction history to a database for audit and performance tracking.

B. Data Flow

The data flow begins with the capture or upload of a traffic sign image. The image is passed to the preprocessing module where it is resized to 50x50 pixels and normalized. The normalized array is expanded to include a batch dimension and fed into the CNN model. The Softmax output layer produces a probability distribution over 43 classes. The class with the highest probability is returned as the predicted label along with the confidence score, which is then displayed on the web interface and logged to the database.

employing transfer learning with VGG16 for multi-class traffic sign classification across all 43 GTSRB categories. A Flask-based web application is integrated to provide real-time image upload and prediction capability, making the system practically deployable in intelligent transportation and autonomous vehicle contexts. The rest of the paper is organized as follows: Section II reviews related literature, Section III states the problem, Sections IV–VI describe the proposed system and implementation, Section VII presents experimental results, and Section VIII concludes the work.

II. LITERATURE SURVEY

Lin (2025) evaluated CNN, ResNet-18, and MobileNetV2 on GTSRB and found that a customized CNN achieved a well-balanced trade-off between accuracy (95.35%) and inference speed, while ResNet-18 achieved slightly higher accuracy at the cost of longer inference time. MobileNetV2 exhibited lower performance despite its lightweight nature [2].

Chen (2025) provided a comprehensive survey of recent advances in traffic sign recognition, covering models such as YOLOv8-TS, STD-YOLOv5s, Semantic Scene Understanding, and Cascade R-CNN with Multi-Scale Attention. The study evaluated precision, recall, F1-score, and mean Average Precision (mAP) across methods, emphasizing robustness to real-time detection and varied lighting conditions [3].

Alawaji, Hedjar, and Zuair (2024) applied multi-task learning (MTL) combining shared CNN features with InceptionResNetV2 and DenseNet201, enhanced by a YOLOv7 detection module for localizing signs in real street scenes. The model achieved high accuracy on both recognition and localization tasks, demonstrating the advantage of shared feature learning under adverse conditions [4].

Yalamanchili et al. (2024) proposed a hybrid system combining YOLOv5 with Autoencoder-CNN, Autoencoder-LSTM, and RNN architectures to improve recognition performance. The study showed that blending object detection with sequential learning techniques significantly enhances robustness and precision across diverse traffic environments [5].

Elside and Abougarair (2025) built a recognition system using both a custom CNN and pre-trained VGG19 on GTSRB, comparing training setups and demonstrating that combining custom architectures with transfer learning achieves reliable classification results [6].

Tiryaki and Oral (2025) enhanced traffic sign classification by integrating Spatial Transformer Networks (STNs) with CNNs to reduce the impact of geometric distortions. Evaluated on the Russian Traffic Signs Dataset (RTSD), the STN-CNN model outperformed conventional CNNs, emphasizing the need to handle geometric variability in real-world images [7].

Hai et al. (2024) performed a comparative analysis of Faster R-CNN, YOLOv8, ResNet50, VGG16, VGG19, and LeNet for TSR. YOLOv8 achieved leading detection performance with 99.4% mAP, while lightweight LeNet offered efficient classification at 98.2% accuracy, providing insights into balancing accuracy against computational cost [8].

Lu et al. (2025) presented a novel framework combining

C. Module Description

The system is organized into four functional modules. The Data Preprocessing Module handles resizing, normalization, and augmentation of input images. The Data Cleaning Module removes duplicate, corrupted, or mislabeled images to maintain dataset integrity. The Model Training Module implements the CNN and VGG16 training pipeline using backpropagation with Adam optimization, monitoring training and validation metrics across epochs. The Prediction Interface Module provides the Flask web application endpoints for user interaction, real-time inference, and scan history retrieval.

D. Technology Stack

The system is built using Python 3.9, TensorFlow 2.x, and Keras for deep learning. Anaconda and Jupyter Notebook serve as the development environment. Supporting libraries include NumPy for array operations, Pandas for data handling, OpenCV for image processing, Scikit-learn for evaluation metrics, Matplotlib and Seaborn for visualization, Flask for the web backend, Flask-SQLAlchemy for database management, Flask-JWT-Extended for authentication, and Werkzeug for secure file handling.

VI. IMPLEMENTATION

The GTSRB dataset is loaded from organized class subdirectories using OpenCV. Each image is resized to 50x50 pixels, converted to a NumPy array, and normalized by dividing pixel values by 255.0. Labels are extracted from directory names and one-hot encoded using Keras to_categorical(). The dataset is then split into training, validation, and test subsets using Scikit-learn's train_test_split() function.

Model training employs Keras ImageDataGenerator for real-time data augmentation. Training and validation accuracy and loss are tracked across epochs and visualized using Matplotlib. After training, the model achieving the best validation accuracy is saved as traffic_sign_model.h5 using Keras ModelCheckpoint callback.

The Flask application initializes by loading the saved model and class index mapping from class_indices.json. The prediction pipeline loads an uploaded image using Keras load_img(), converts it to a NumPy array, normalizes it, and calls model.predict() to obtain class probabilities. The index of the highest probability is mapped to the GTSRB class label and returned to the user along with confidence percentage. Each successful prediction is saved to the History table in the SQLite database linked to the authenticated user.

Confusion matrices and per-class classification reports are generated using Scikit-learn to evaluate model performance across all 43 sign categories. These metrics help identify frequently misclassified sign pairs, which are used to guide targeted data augmentation in subsequent training iterations.

VII. TESTING AND RESULTS

A. Testing Methodology

Five complementary testing strategies were employed to validate the system comprehensively. Unit Testing verified individual modules—preprocessing, augmentation, model inference, and database operations—in isolation. Integration Testing validated the complete pipeline from image upload through preprocessing, prediction, and database logging. Functional Testing verified that all system functions—signup, login, upload, predict, and history retrieval—behaved as

open-vocabulary detection with contrastive learning and multimodal fusion of visual and semantic text features. On the TT100K benchmark, the method achieved state-of-the-art detection and classification by effectively mitigating class imbalance and scale variation [9].

Sah et al. (2025) integrated ResNet-50, YOLOv8, and RT-DETR with multimodal large language models for comprehensive traffic sign recognition and lane detection. ResNet-50 achieved near-perfect accuracy, and the multimodal framework enhanced autonomy in adverse driving conditions, demonstrating the potential of combining vision and language models [10].

III. PROBLEM STATEMENT

Manual traffic sign recognition by human drivers is inherently prone to errors caused by distractions, fatigue, poor visibility, and adverse environmental conditions such as fog, rain, and low lighting. In autonomous driving systems, accurate and real-time recognition is essential for safe navigation and decision-making; failure to correctly interpret signs can result in serious accidents and traffic violations.

A major challenge in building a reliable TSR system is constructing large, diverse, and well-labeled datasets. Traffic signs may appear in different sizes, orientations, and occlusion levels. Small or imbalanced datasets often lead to overfitting, where the model performs well during training but fails to generalize to unseen images. Additionally, environmental factors such as faded signs, motion blur, shadow effects, and cluttered backgrounds further increase classification complexity.

Although deep CNNs deliver high accuracy, deeper architectures demand significant computational resources and careful hyperparameter tuning. Deploying such models in real-time embedded systems (e.g., ECUs in autonomous vehicles) requires optimizing inference speed without sacrificing accuracy. Therefore, achieving an optimal balance between classification performance, computational efficiency, and real-world robustness remains an open and critical research challenge.

IV. PROPOSED SYSTEM

The proposed system introduces a deep learning-based approach using Convolutional Neural Networks and transfer learning with VGG16 for accurate and robust traffic sign recognition. Unlike traditional methods, the model automatically extracts relevant features directly from raw images without relying on manual feature engineering. The end-to-end pipeline consists of data preprocessing, CNN model training, performance evaluation, and web-based deployment.

A. Dataset

The German Traffic Sign Recognition Benchmark (GTSRB) is used as the primary dataset. It contains 51,839 labeled images distributed across 43 traffic sign classes including speed limits, prohibitory signs, danger signs, mandatory signs, and directional signs. Images vary in resolution, lighting, and quality to simulate real-world conditions. The dataset is split into 70% for training (36,287 images), 15% for validation (7,776 images), and 15% for testing (7,776 images).

specified. Accuracy Testing evaluated model performance on the held-out test set using standard classification metrics. Performance Testing measured response time and throughput under single and concurrent user loads to confirm real-time suitability.

B. Quantitative Results

Table I presents the performance comparison between the custom CNN and VGG16 transfer learning model on the GTSRB test set of 7,776 images.

Metric	Custom CNN	VGG16 + TL
Accuracy	95.4%	97.8%
Precision	94.8%	97.2%
Recall	94.5%	97.0%
F1-Score	94.6%	97.1%
Inference Time	12 ms	18 ms
Training Time	~22 min	~38 min

Table I. Performance Comparison on GTSRB Test Set

VGG16 with transfer learning outperforms the custom CNN across all classification metrics, achieving 97.8% accuracy and 97.1% F1-score. Although VGG16 has a slightly longer inference time (18 ms vs. 12 ms), both models satisfy real-time requirements for autonomous driving applications typically requiring responses within 100 ms.

C. Robustness Evaluation

To evaluate robustness, the test set was augmented with three types of distortions: Gaussian noise (sigma=0.05), motion blur (kernel size 5x5), and brightness variation (+/-40%). Under these conditions, VGG16 maintained an average accuracy of 92.3%, compared to 87.6% for the custom CNN. The largest degradation occurred under heavy motion blur for both models, but VGG16's pre-trained feature representations proved significantly more resilient, confirming the advantage of transfer learning for real-world deployment.

D. Comparison with Existing Methods

Table II compares the proposed system with recent state-of-the-art TSR methods reported in the literature on the GTSRB dataset.

Method	Architecture	Accuracy
Lin [2], 2025	Custom CNN	95.35%
Hai [8], 2024	LeNet	98.2%
Elside [6], 2025	VGG19 + TL	96.7%
Proposed	VGG16 + TL	97.8%

Table II. Comparison with Existing TSR Methods

The proposed VGG16 transfer learning approach achieves competitive accuracy among existing methods. While Hai et al.'s LeNet achieves 98.2% with lower computational requirements, the proposed system offers a well-rounded balance of accuracy, robustness, and practical deployability via its integrated web application.

B. Data Preprocessing and Augmentation

All input images are resized to 50x50 pixels and pixel values are normalized to the range [0, 1] to ensure uniform input distribution. Data augmentation is applied using Keras ImageDataGenerator with the following transformations: rotation up to 15 degrees, width and height shifts up to 10%, zoom range of 10%, and brightness adjustment in the range [0.8, 1.2]. These augmentations simulate real-world variations in sign orientation and lighting, significantly improving model generalization and reducing overfitting on the training set.

C. CNN Architecture

The custom CNN architecture comprises three convolutional blocks followed by fully connected layers. Block 1 applies 32 filters (3x3, ReLU) followed by MaxPooling (2x2). Block 2 applies 64 filters (3x3, ReLU) with MaxPooling. Block 3 applies 128 filters (3x3, ReLU) with MaxPooling. The feature maps are then flattened and passed through a Dense layer (256 neurons, ReLU) with a Dropout layer (rate=0.5) to prevent overfitting. The output layer uses 43 neurons with Softmax activation for multi-class probability distribution.

For the transfer learning variant, VGG16 pre-trained on ImageNet is used as the feature extractor. The convolutional base is frozen during initial training, and custom classification layers—Dense (512, ReLU) + Dropout (0.5) + Dense (43, Softmax)—are appended and trained. In the fine-tuning phase, the last four convolutional layers of VGG16 are unfrozen and trained with a reduced learning rate of 1e-5 to adapt ImageNet features to traffic sign characteristics.

D. Training Configuration

Both models are compiled using the Adam optimizer with categorical cross-entropy loss. The custom CNN is trained for 30 epochs with an initial learning rate of 0.001 and batch size of 64. VGG16 fine-tuning uses a learning rate of 1e-5 for 20 additional epochs. Early stopping (patience=5 epochs) and ReduceLROnPlateau (factor=0.5, patience=3) callbacks are used to prevent overfitting and improve convergence. The best model weights are saved as traffic_sign_model.h5 for deployment.

VIII. APPLICATIONS AND FUTURE SCOPE

A. Applications

- Autonomous vehicles for real-time sign detection, interpretation, and speed regulation
- Advanced Driver Assistance Systems (ADAS) for auditory and visual sign-based alerts to drivers
- Smart city traffic management platforms for automated sign monitoring and compliance enforcement
- Mobile-based road safety applications for cyclists, pedestrians, and learner drivers
- Embedded roadside systems for intelligent transportation infrastructure and sign maintenance detection

B. Future Enhancements

Several enhancements are planned for future iterations of this system. First, the model can be extended to support traffic sign datasets from multiple countries and regions, improving global applicability. Second, real-time integration with live camera feeds using OpenCV will enable continuous video-based recognition, making it suitable for direct autonomous vehicle integration. Third, lightweight model variants such as MobileNetV3 and EfficientNet-Lite can be explored for deployment on low-power embedded hardware such as the NVIDIA Jetson Nano or Raspberry Pi. Fourth, attention-based mechanisms and Vision Transformers (ViTs) can be investigated to further improve recognition of small or partially occluded signs. Finally, a federated learning approach may allow the model to continuously improve using data from distributed vehicle fleets without compromising data privacy.

IX. CONCLUSION

This paper presented a deep learning-based Traffic Sign Recognition System using custom CNN and VGG16 transfer learning trained on the GTSRB dataset containing 43 sign categories. Extensive image preprocessing and data augmentation were applied to improve model generalization. The VGG16 transfer learning model achieved 97.8% classification accuracy, 97.2% precision, 97.0% recall, and 97.1% F1-score on the held-out test set, outperforming the custom CNN baseline across all metrics. Robustness evaluation under noise, blur, and lighting variation confirmed VGG16's superior adaptability to real-world conditions. A Flask-based web application with JWT authentication, real-time image upload, and scan history logging was implemented to demonstrate practical deployability. Comparative analysis showed that the proposed system performs competitively against recent state-of-the-art methods. This work highlights the practical effectiveness of deep learning and transfer learning in building reliable, scalable, and deployable traffic sign recognition systems for intelligent transportation and autonomous driving applications.

-

REFERENCES

[1] Y. Zhang and H. Liu, Multi-Scale CNN for Real-Time Traffic Sign Recognition, IEEE Trans. Intelligent Transportation Systems, vol. 26, no. 2, pp. 1345-1358, 2025.

[2] T. Lin, Comparative Analysis of Deep Learning Models for Efficient Traffic Sign Recognition, Applied and Computational Engineering, 2025.

[3] X. Chen, Technological Evolution of Traffic Sign Recognition, Food Science Journal, 2025.

[4] K. Alawaji, R. Hedjar, and M. Zuair, Traffic Sign Recognition Using Multi-Task Deep Learning for Self-Driving Vehicles, MDPI, 2024.

[5] S. Yalamanchili et al., Optimizing Traffic Sign Detection and Recognition by Using Deep Learning, acadlore.com, 2024.

[6] W. E. Elside and A. J. Abougarair, Traffic Sign Recognition Using CNN, 2025.

[7] B. Tiryaki and E. A. Oral, Traffic Sign Classification Using Spatial Transformer Networks (STN)-Based CNN, acadlore.com, 2025.

[8] B. D. Hai et al., Performance Comparison in Traffic Sign Recognition Using Deep Learning, EUDL, 2024.

[9] Q. Lu et al., Contrastive Learning-Driven Traffic Sign Perception: Multi-Modal Fusion of Text and Vision, arXiv, 2025.

[10] C. K. Sah et al., Advancing Autonomous Vehicle Intelligence: Deep Learning and Multimodal LLM for TSR, 2025.