

Traffic Sign Recognition using CNN (Convolutional Neural Network)

N. Shiva Ganesh¹, M. Bharadwaj², M. S. S. Kalyan³, M. Yaswanth⁴, Dr. K. Kishore Raju⁵

U.G. Student, Department of Information Technology^{1,2,3,4}

Associate Professor, Department of Information Technology⁵

S.R.K.R. Engineering College (A), Bhimavaram, Andhra Pradesh, India







Abstract: Effective recognition of traffic signs systems are essential for traffic management since road safety is a major problem. In this work, we introduce a unique method for computer vision and deep learning-based real-time traffic sign identification and categorization. Convolutional neural networks (CNNs), trained on an extensive collection of photographs of traffic signs, are the tool we use in our approach. We improve the quality of the input photos and extract relevant features by preprocessing them, which includes grayscale conversion and normalization. After that, the CNN model examines these characteristics to precisely determine the kind of traffic sign that is shown. To further improve classification accuracy, we also use bounding box localization to correctly outline and extract indicators. The effectiveness of our system is shown by experimental findings, which show that it can reliably detect traffic signs in a variety of environmental conditions. This study provides a reliable and effective method for identifying traffic signs, promoting road safety.

Keywords: Traffic sign recognition, convolutional neural networks, deep learning, bounding box localization

I. INTRODUCTION

Recognizing traffic signs is considered to be essential for maintaining road safety and effective traffic control systems. Traffic sign detection and interpretation automated systems have become indispensable due to the increasing number of cars on the road. Conventional techniques for identifying traffic signs, which frequently rely on rule-based algorithms or manual inspection, may not be scalable and may be prone to mistakes. Recent developments in deep learning—more specifically, in the area of convolutional neural networks (CNNs)—have revolutionised computer vision by making tasks involving object detection more precise and effective. Traffic sign identification using CNNs has shown encouraging results, providing high accuracy real-time detection capabilities. [8].

Table 1: Predefined Subsets in GTSRB

(a) Speed Limit Signs	
(b) Other Prohibitory Signs	
(c) Derestriction Signs	
(d) Mandatory Signs	
(e) Danger Signs	
(f) Unique Signs	

The goal of this study is to use CNNs to construct a real-time traffic sign identification system. The system preprocesses traffic sign images, trains a CNN model using cutting-edge architectures, and integrates it into a real-time detection pipeline by utilizing the German Traffic Sign Recognition Benchmark (GTSRB) dataset [7,11], as indicated in table 1. The system's objective is to precisely identify and categorize traffic signs from live video feeds that are

recorded by a camera by utilizing computer vision and deep learning techniques. The traffic sign recognition system's implementation details, including data collection, model architecture, training procedures, evaluating metrics, are provided in this work. Furthermore, the suggested system's possible uses, restrictions, and future possibilities are examined, emphasizing how important it is to improve traffic control and road safety.

II. LITERATURE SURVEY

The project presents a robust Traffic Sign Recognition (TSR) system utilizing CNNs and Keras for high accuracy and minimal computing time, vital for real-time applications like autonomous driving. By preprocessing a traffic sign dataset and constructing a CNN model, the system detects and classifies signs effectively. Integration of a Tkinter-based graphical user interface improves usability, yet challenges remain, such as reliance on labeled datasets, performance variability, and computational demands, requiring further refinement for practical deployment. [1].

The project presents a data-driven system for recognizing all traffic sign categories in video sequences captured by car-mounted cameras. It comprises three stages: traffic sign regions of interest (ROIs) extraction, ROIs refinement and classification, and post-processing. Maximally Stable Extremal Regions (MSERs) extract ROIs, while a multi-task CNN refines and classifies them. Trained on a large dataset with real and synthetic images, the system achieves state-of-the-art results on a challenging dataset, demonstrating real-world effectiveness. Future work includes automatic labelling of additional data, end-to-end detection frameworks, and improving system speed and post-processing. [2].

The research focuses on a Traffic Sign Recognition (TSR) system tailored for Bangladeshi traffic signs, using color cues and CNNs. It involves image acquisition, preprocessing, segmentation, and automatic feature extraction and classification. Emphasis is on triangular signs with red rims prevalent in Bangladesh. The proposed algorithm achieves good recognition accuracy, addressing country-specific sign characteristics. Challenges include hardware limitations during training, with future work aimed at enhancing speed, expanding detection capabilities, and implementing real-time functionality [3].

The project delves into traffic sign recognition (TSR) for autonomous driving using contemporary methods and datasets from Belgium and Germany. While existing methodologies excel, leveraging features like Histogram of Oriented Gradients (HOG) and sparse representations, they encounter limitations due to dataset saturation. The absence of adverse weather datasets impedes comprehensive evaluation and robustness enhancement. In summary, the project emphasizes the necessity for ongoing innovation and dataset expansion to advance TSR for real-world applications. [5].

The project presents a novel approach to real-time traffic sign recognition, combining Faster R-CNN and Mobile Nets frameworks to achieve superior detection accuracy on the GTSDB database. An efficient CNN classifier further enhances performance, enabling real-time applications. However, challenges persist in improving recall rates and robustness under diverse conditions, requiring future research efforts to address these limitations [7][6].

The project develops a CNN-based traffic sign detection and classification system for ADAS, showing competitive performance on the BTSD dataset. Despite effectively addressing real-world challenges like color fading and occlusion, scalability and generalization limitations persist due to dataset constraints. Further optimization is needed to improve robustness and handle diverse traffic sign images encountered in different environmental conditions. Nonetheless, the system advances traffic sign detection technology, with potential benefits for road safety and transportation efficiency [10].

III. PROPOSED SYSTEM

Traffic sign recognition plays a crucial role in intelligent transportation systems, enhancing road safety and traffic management. In this paper, we propose a real-time traffic sign recognition system using [1][9] Convolutional Neural Networks (CNNs) shown in fig 1. Our proposed system aims to accurately detect and classify traffic signs in real-time video streams captured by a camera mounted on vehicles or traffic surveillance systems. The system can be broken down into following stages:

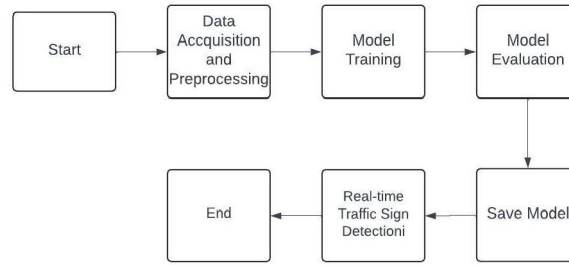


Fig 1: Traffic Sign Recognition System Flow Chart

1. Data Acquisition and Preprocessing: We begin by acquiring a dataset containing images of traffic signs. In our research, we utilize the German Traffic Sign Recognition Benchmark (GTSRB) dataset shown in table 1. Each image is preprocessed to standardize its size, typically resizing to 32x32 pixels, and undergoes normalization to enhance model performance [4].

2. Convolutional Neural Network Architecture: CNNs are employed for their effectiveness in image classification tasks. The architecture consists of multiple layers, including convolutional layers, pooling layers, and fully connected layers. The mathematical functions involved in building the CNN model include:

Convolutional Operation:

$$y(i,j) = \sum_m \sum_n x(i+m,j+n) \times w(m,n) + b \quad \text{-(Eq1)}$$

where,

i and j represents spatial indices of the output feature map ‘y’.

x represents the input image,

w denotes the filter weights,

y (i, j) represents the output feature map at position (i, j)

x (i+m, j+n) represents the pixel value of the input image at position (i+m, j+n)

w (m, n) represents the filter weight at position (m, n)

b is the bias term.

Activation Function (ReLU): Rectified Linear Unit introduces non-linearity to the model.

$$f(x) = \max(0, x) \quad \text{-(Eq.2)}$$

f(x) represents the output of the activation function

x represents the input value

max(0, x) returns the maximum of 0 and the input value x, introducing non-linearity.

Pooling Operation (Max Pooling):

$$y(i, j) = \max(x(2i,2j), x(2i,2j+1), x(2i+1,2j), x(2i+1,2j+1)) \quad \text{-(Eq.3)}$$

y (i, j) represents the output of the pooling operation at (i, j)

i and j represent the indices of the pooled regions in feature maps.

reducing the spatial dimensions of the feature maps.

Fully connected Layer: Each neuron is connected to every neuron in the previous and next layers.

Model Training:

Forward Pass: Input images from the GTSRB dataset containing images of traffic signs. In our research, we utilize the German Traffic Sign Recognition Benchmark (GTSRB) dataset shown in the table 1. Each image is preprocessed to standardize its size, typically resizing to 32x32 pixels, and undergoes normalization to enhance model performance [4].

Calculate the Loss: The categorical cross-entropy loss function evaluates the disparity between predicted and actual class labels for each input image, facilitating multi-class classification.

Backpropagation: Gradients of the loss function with respect to model parameters are computed through backpropagation. This involves propagating gradients backward through the layers, enabling adjustments to the model’s weights.

Update Weights: The Adam optimizer iteratively updates model weights based on computed gradients, minimizing the loss function and enhancing the model's ability to accurately classify traffic signs.

Conclusion: The training process optimizes model parameters to minimize loss, enabling effective extraction of features from input images and accurate classification of traffic signs, vital for real-time applications.

Data Augmentation

To increase the diversity of the training dataset and improve the model's generalization ability, data augmentation techniques such as width and height shifting, zooming, shearing, and rotation are applied.

Table 2: Augmentation Parameters

Parameter	Value
width_shift_range	0.1
height_shift_range	0.1
zoom_range	0.2
shear_range	0.1
rotation_range	10

Evaluation and Testing

The process of evaluating the trained model involves utilizing a separate test dataset to gauge its performance. Various metrics such as accuracy, precision, recall, and F1-score are computed to provide a comprehensive assessment of the model's effectiveness.

For classification tasks, the output of the Convolutional Neural Network (CNN) model is categorized as follows: Firstly, the input image undergoes preprocessing, typically resizing to conform to the model's input dimensions. The preprocessed image is then fed into the trained CNN model using the **model.predict** function, resulting in a probability distribution across different output classes. Subsequently, the class with the highest probability is chosen as the predicted class for the input image. This selection process is facilitated by the **np.argmax** function, which identifies the index corresponding to the maximum probability in the probability array. Once the index of the predicted class is determined, it is mapped to the corresponding traffic sign class label through a function called **getClassName()**, making the interpretation human-readable.

To refine prediction accuracy, a probability threshold can be applied to filter out low-confidence predictions. By setting a threshold value (example., threshold = 0.75), predictions with probabilities below this threshold are deemed unreliable and thus excluded from further consideration. Finally, the predicted class label, along with the corresponding probability value, is overlaid onto the input image using **OpenCV's cv2.putText** function [7]. These steps collectively enable accurate classification and visualization of traffic sign classes in real-world scenarios, facilitating efficient interpretation and identification of traffic signs.

These steps collectively ensure accurate classification and visualization of traffic sign classes in real-world scenarios, thereby facilitating efficient interpretation and identification of traffic signs.

Loss Function: The loss function employed in our proposed system is categorical cross-entropy, which is specifically designed for multi-class classification tasks. This loss function quantifies the disparity between the predicted class probabilities and the actual class labels. Mathematically, categorical cross-entropy is defined as:

$$\text{Categorical cross-entropy loss} = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^M y_{ij} \log(p_{ij}) \tag{Eq.4}$$

N is the number of samples

M is the number of classes

y_{ij} is a binary indicator of whether class j is the correct classification for sample i.

p_{ij} is the predicted probability that sample i belongs to class j

Deconstructing the model:

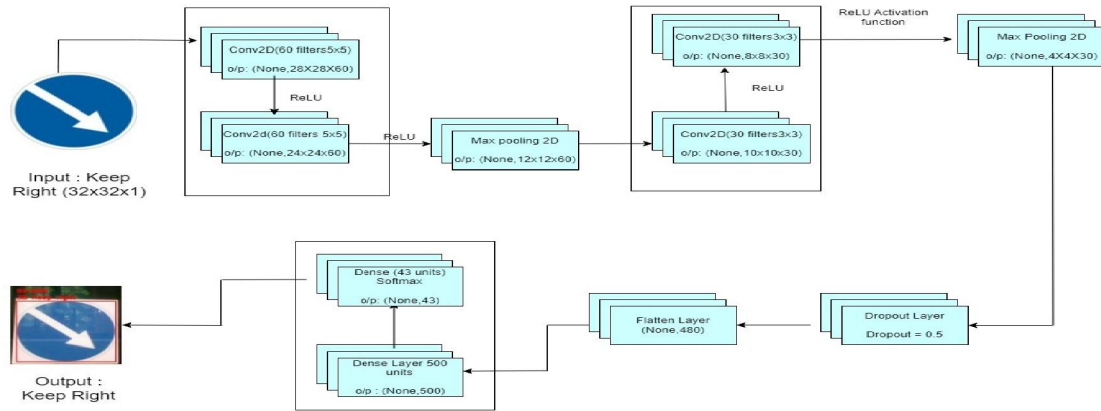


Fig2: CNN Model Architecture

A closer examination of the model architecture reveals a series of intricate operations meticulously executed at each layer. The first convolutional layer, in the Fig2, conv2d, applies 60 filters of size (5, 5) to the input images, extracting basic features such as edges and textures. Each filter produces a feature map, resulting in output feature maps with dimensions (None, 28, 28, 60), where None indicates the batch size. Following this, conv2d_1 applies another set of 60 filters of size (5, 5) to capture more complex patterns and features from the input images, resulting in output feature maps with dimensions (None, 24, 24, 60) as shown in fig 2.

The max_pooling2d layer then performs max pooling operations on the feature maps obtained from the previous convolutional layers, reducing the spatial dimensions by a factor of 2 with a pool size of (2, 2). This down sampling results in output feature maps with dimensions (None, 12, 12, 60). Subsequently, conv2d_2 as mentioned in fig2, applies 30 filters of size (3, 3) to extract specific and higher-level features from the reduced feature maps, yielding output feature maps with dimensions (None, 10, 10, 30). Conv2d_3 further refines the extracted features with another set of 30 filters of size (3, 3), resulting in output feature maps with dimensions (None, 8, 8, 30).

The max_pooling2d_1 layer performs another max pooling operation with a pool size of (2, 2), further reducing the spatial dimensions of the feature maps to (None, 4, 4, 30). A dropout layer with a rate of 0.5 is utilized for regularization purposes, randomly deactivating 50% of the neurons in the previous layer during training to prevent overfitting. Following this, the flatten layer reshapes the output from the previous layer into a 1D array, preparing the data for input to the fully connected layers by converting the output dimensions to (None, 480).

A dense layer with 500 neurons allows the model to learn complex patterns and relationships in the flattened input data, with output dimensions of (None, 500). Finally, the output layer, dense_1, consists of 43 neurons corresponding to the number of classes in the dataset. This layer applies softmax activation to produce class probabilities for each input image, with output dimensions of (None, 43).

IV. EXPERIMENTAL RESULTS

In this research, the proposed model was trained using VS Code, a robust development environment well-suited for machine learning endeavors. The dataset consisted of a total of 51,840 images, partitioned into a training set comprising 39,209 images and a test set containing 12,631 images, all in PNG format. Before being fed into the model, the images were resized to dimensions of 32x32x1. Each image was annotated with bounding box coordinates (Roi.X1, Roi.Y1, Roi.X2, Roi.Y2) and class labels (ClassId). Additionally, the dataset provided corresponding .csv files containing metadata such as image dimensions (Width, Height) and file paths. These images underwent processing and analysis on a system operating on Windows 11, equipped with suitable hardware configurations. The superior configuration contributed to achieving commendable results, including accurate image predictions and high model accuracy.

Visualizations, such as the loss curve and accuracy plot, provided valuable insights into the training dynamics of the model. The loss curve (depicted in fig:4) showcased a consistent decline in both training and validation loss across

epochs, indicating the model's adeptness at minimizing errors during training. Concurrently, the accuracy plot (illustrated in fig:3) portrayed a steady upward trend in both training and validation accuracy throughout the training phase, underscoring the model's progressively improving capability to correctly classify images with each iteration.

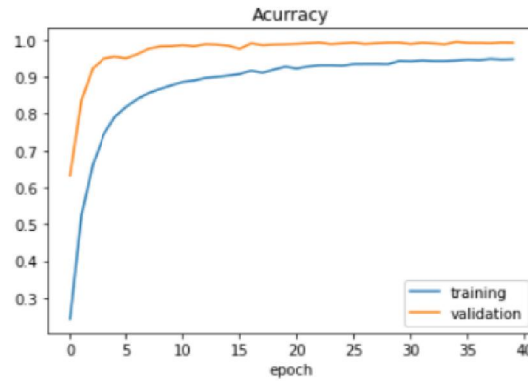


Fig 3: Training and validation curve for accuracy w.r.t epochs

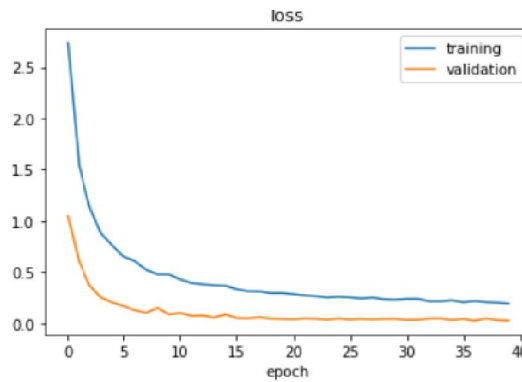


Fig 4: Training and validation curve for loss w.r.t epochs

The confusion matrix delivered an in-depth analysis of the classification performance of the model. Notably, the high values along the diagonal of the matrix depicted in Fig:5 prove that a majority of images were correctly classified.

Confusion Matrix:

$$\begin{bmatrix}
 34 & 1 & 0 & \dots & 0 & 0 & 0 \\
 0 & 431 & 2 & \dots & 0 & 0 & 0 \\
 0 & 0 & 490 & \dots & 0 & 0 & 0 \\
 \dots & & & & & & \\
 0 & 0 & 0 & \dots & 67 & 0 & 0 \\
 0 & 0 & 0 & \dots & 0 & 44 & 0 \\
 0 & 0 & 0 & \dots & 0 & 0 & 53
 \end{bmatrix}$$

Fig 5: Confusion Matrix

After analyzing the performance metrics of the existing system, it was concluded that the proposed model exhibits high accuracy, precision, and recall. Table 3 indicates that the accuracy of the system is 95.16%, precision is 95.27%, recall is 95.16%, and F1-score is 95.06%. Despite these high metrics, a few misclassifications were observed across different classes, suggesting room for improvement. Overall, the trained convolutional neural network (CNN) model

demonstrated exceptional performance across various evaluation metrics. It achieved an impressive accuracy of 99.08%, indicating its proficiency in correctly classifying images into their respective classes. Precision, recall, and F1 score further confirmed the model's robustness in classification tasks, with precision at 99.05%, recall at 99.08%, and an F1 score of 99.02%, as depicted in Table-3.

Table 3: Performance Metrics

Metrics	CNN Model using image as input	CNN Model using frames as input (Proposed model)
Precision	95.27%	99.05%
F1-score	95.06%	99.02%
Recall	95.16%	99.08%
Accuracy	95.16%	99.08%

The model accurately predicted various traffic signs with high confidence levels. For instance, the yield sign was identified with 100.0% certainty, while the bumpy road sign and turn left ahead sign were predicted with 99.59% and 99.89% confidence, respectively. Additionally, the children crossing sign was recognized with 92.45% accuracy. These results underscore the model's effectiveness in real-world applications, particularly in traffic sign recognition tasks.



(a) Yield sign



(b) Bumpy Road sign



(c) Turn left ahead sign



(d) Children crossing sign

Fig 5: Different traffic sign images

V. CONCLUSION

The model proposed has been implemented successfully and achieved the desired results. The model achieved prominent results in comparison with existing system which made an accuracy of 95%. The CNN model architecture demonstrates strong performance in image classification, achieving noteworthy accuracy, precision, recall, and F1 score metrics shown in table 3. The model has been implemented in real time wherein, the inputs are captured as frames

showcasing prominent results through model prediction shown in fig 5. The traffic sign images are predicted with high accuracy. The CNN model achieved an accuracy of 99.08% with an increase of 4% more than the existing system. To improve transportation efficiency and safety, traffic sign recognition technologies will eventually be integrated into autonomous vehicles. Enhancing real-time detection accuracy, maximizing performance in a range of scenarios, and enabling smooth interaction with car infrastructure are the objectives. Plans call for adding sensor fusion methods, extending into dynamic sign identification, and putting continuous learning mechanisms in place. To standardize and encourage broad use, cooperation with stakeholders is necessary to advance autonomous driving technology

REFERENCES

- [1] Raghul B, Sakthivel K, Raghul E (2022). Traffic sign Recognition using CNN and Keras. Internation
- [2] Hengling Luo, Yi Yang, et al (2018). TSR using a Multi-Task CNN. IEEE Transactions on Intelligent Transportation Systems. IEEE Xplore.
- [3] Prashengit Dhar, Md.Zainal Abendin (2017). TSR- A New Approach and Recognition Using CNN. IEEE Xplore.
- [4] Xuehong Mao, Samer Hijazi (2016), et al. Hierarchial CNN for Traffic Sign Recognition. IEEE Intelligent Vehicles Symposium (IV). IEEE Xplore.
- [5] Markus Mathias, Radu Timofte, Traffic Sign Recognition (2013) . How far we are from the solution? IEEE Xplore.
- [6] Jia Li and Zengfu Wang (2019). Real-Time Traffic Sign Recognition Based on Efficient CNN's in the Wild. IEEE Xplore.
- [7] Hee Seok and Kang Kim (2018). Simultaneous Traffic Sign Detection and Boundary Estimation Using and CNN. IEEE Xplore.
- [8] Abhay Lodhi, Sagar Singhal, Massoud Massoudi (2021). Car Traffic Sign Recognizier using CNN. IEEE Xplore.
- [9] Syed Hussian, Munther Abdulkibash, Samir Tout (2018). A Survey of Traffic Sign Recognition Systems Based on Convolutional Neural Networks. IEEE Xplore.
- [10] Smit Mehta, Chirag Paunwala, Bhaumik Vaidya (2019). CNN Based Traffic Sign Classification using Adam Optimizer. Intelligent Computing and Control Systems (ICICCS).
- [11] Zhilong, He Zhogjun, Xiao (2021) Traffic Sign Recognition Based on CNN. IEEE Xplore