Traffic Sign Detection & Classification for Autonomous Vehicles

Kalyan Roy Veerabhadra Rao Marellapudi Vishwas Bhushan Basuru Mohammed Uvais
Yeshiva University, NYC, NY
kroy@mail.yu.edu, vmarella@mail.yu.edu, vbasuru@mail.yu.edu, ptuvais@gmail.com

Abstract

Deep learning and automotive advancements are driving the development of smart cars and autonomous vehicles. Traffic sign recognition, essential for safe autonomous driving, is being tackled with innovative deep-learning techniques. These signs are paramount for global traffic flow and driver safety. Recognizing them improves driver assistance systems and the driving experience in general. While challenges like weather variations and sign degradation exist, Smart Driving Support Systems offer promising solutions for both Advanced Driver-Assistance Systems and future autonomous vehicles.

*Our research introduces two novel CNN architectures, namely the **TSNet** and **TSDetector** CNN models, trained on the 'German Traffic Sign Detection Benchmark' (GTSDB) dataset. The goal is to improve traffic sign detection and classification accuracy and contribute to the progression of autonomous vehicle technology, ultimately mitigating traffic accidents.*

In the object detection task, our custom TSDetector model outperformed both FasterRCNN and RetinaNet. For classification, the pre-trained ResNet50 model achieved the highest accuracy, surpassing MobileNetV2 and the custom TSNet. Our aspiration is for these models to surpass traditional methods, thereby further unlocking the potential of custom deep-learning architectures for real-world autonomous driving applications.

Keywords: *Deep Learning, Traffic Sign Recognition, Autonomous Vehicles, CNN, GTSDB.*

1. Introduction

The advancement of technology in Advanced Driving Assistance Systems (ADAS) is continuously evolving. Object detection along with Image Classification plays a crucial role in enhancing Intelligent Driving and Traffic Safety, particularly in the emerging era of autonomous vehicles [3]. Object detection is a core computer vision task that underpins complex functionalities like object tracking and anomaly detection. Convolutional Neural Networks (CNNs) have achieved remarkable success in object detection, prompting researchers to apply them to traffic sign data using two main approaches: single-stage and two-stage detection.

Tesla Inc. stands out as the foremost manufacturer of self-driving cars globally and achieved the status of the most valuable car company during the drafting of this document, indicating the significant demand for autonomous vehicles in the automotive industry. The capability of autonomous vehicles to accurately detect, classify, and respond to various traffic signs holds immense importance for human safety, necessitating precision and speed in detection technology. Given the potential of such systems to save lives and reduce costs, the development and enhancement of such technology serve as primary motivations for this study. Our primary objective is to utilize deep learning algorithms to devise a system capable of addressing these challenges.

A notable advancement in scene understanding is OLIMP, introduced by Amira et al. (2020) [2], which leverages multimodal datasets for enhanced environmental perception. The diverse shapes and color combinations aid in distinguishing between different signs. However, despite various characteristics that differentiate traffic signs, their recognition remains a formidable task due to factors such as color variation, shape complexity, environmental conditions, occlusion, and varying illumination levels [6]. Several classes of low-light image enhancement algorithms, along with their improved versions, have been proposed to mitigate illumination issues in images.

According to the World Health Organization (WHO), implementing speed regulations on roads is crucial for reducing traffic accidents. Nevertheless, many drivers frequently surpass these limits, increasing the risk of potential dangers. Road accidents often arise due to various factors such as inadequate signage quality, driver distraction, or obstacles obstructing sign visibility, all of which are typically attributed to human errors. A proficient traffic sign recognition system needs to effectively identify signs in images or video footage captured by cameras and subsequently classify these visuals. In the realm of traffic sign detection, the

algorithm should be flexible in scale, as the distance between the capturing device and the vehicle may vary. Automating the process of traffic sign detection could help alleviate such errors and accurately pinpoint road signs.

The ongoing study contributes to this cause, examines various classification and detection approaches, and aspires us to introduce the TS-Net and TSDetector CNN models. Additionally, we have conducted a comparative study using traditional deep learning models to detect traffic sign images from the publicly available 'German Traffic Sign Detection Benchmark' (GTSDB) dataset [6]. By employing custom-built models and pre-trained models, the study aims to accurately construct a deep learning architecture capable of detecting and recognizing traffic signs. This research is envisioned to make valuable contributions to the automotive industry and innovation in autonomous driving technology, furthering the collective goal of promoting global traffic management and aiding driver communication and safety. Building upon existing work, we have taken these strides to further the study.

- Construct a bespoke TS-Net and TSDetector Convolutional Neural Network (CNN) model utilizing a substantial data set of the "GTSDB dataset" and perform a comparative analysis against pre-trained models from contemporary studies. The objective is to discern and scrutinize potential research gaps in the existing research papers.

- Furthermore, the proposed CNN model will undergo evaluation using established metrics to gauge its performance.

2. Related Work

In academic and industry circles, significant efforts have been directed towards developing adaptable traffic signal control systems. Recently, there has been a notable rise in the use of deep learning (DL) for image classification and detection, drawing attention from various quarters. Various studies have leveraged a range of DL techniques to construct detection models using image data. Convolutional Neural Networks (CNNs) stand out for their effectiveness in real-time image detection and recognition, rendering them especially advantageous in domains such as autonomous vehicles. Their applications span image recognition, object detection, facial recognition, and medical image analysis.

Considerable research has been dedicated to the identification of traffic and road signs. Akatsuka and Imai [1] pioneered the field with their 1987 study on "Traffic Sign Recognition," aiming to develop a rudimentary system capable of identifying traffic signs, notifying drivers, and ensuring safety. However, their efforts were limited to automating the identification of a subset of traffic signs.

Pandey and colleagues (Pandey et al. [11]) employ the Template Matching technique, specifically `CCORR_NORMED`, to recognize traffic signs. Their proposed method consists of two main phases: training and testing. During training, signs from Indian Regulatory Road Signs (IRRS) are used to educate the system. A template is then generated by directly selecting and capturing the road sign, defining it as a Region of Interest (ROI), and storing it in the database along with its assigned name. Additionally, if the image quality is compromised due to lighting variations during template creation, Histogram Equalization is applied to enhance color consistency. In the testing phase, the source image is systematically compared with the template image by sliding it pixel by pixel from left to right and top to bottom. Matching pixel values are recorded in a matrix, and upon obtaining the matrix with maximum intensity, a speech trigger and a display message are activated. While their system demonstrates proficiency in identifying traffic signs under various lighting conditions, it struggles with detecting signs from low-quality, blurred images as aligning intensities during template matching proves challenging. Moreover, due to the inherent nature of Template Matching, instances of false positives—indicating the presence of a sign where none exists in the image—are also observed, diminishing the system's reliability.

Shangzheng [13] introduced a rapid traffic sign recognition approach utilizing an enhanced CNN model integrated with image processing, machine vision techniques, and deep learning for target classification. Their method employs HOG features for traffic sign detection and an upgraded CNN model for precise sign determination. This model significantly enhances the accuracy of traffic sign localization while also reducing false detection rates.

Tsoi and Wheelus [15] utilized a cost-sensitive deep learning CNN approach for Traffic Sign Recognition (TSR), achieving an accuracy of 86.5%. Their research involved constructing a model for traffic signal classification, employing techniques such as cost-sensitive rejection sampling and utilizing cost-sensitive loss functions. The baseline architecture of their model included 3 convolution layers, 2 max-pooling layers, 3 fully connected layers, 2 dropout layers, and a softmax layer.

Lee et al. [9] introduced a system for identifying traffic signs that simultaneously determines their location and precise boundaries using a Convolutional Neural Network (CNN). The study frames the estimation of traffic sign borders as a problem of predicting 2D pose and shape classes, successfully addressed by a single CNN. The authors achieved this by projecting the boundary of a template sign image onto the input image plane based on the expected 2D posture and shape class of the target traffic sign in the input image, thereby approximating its actual boundary.

Islam [7] employed the LeNet framework to categorize traffic signs. The author introduces two CNN-based models: one for sign classification and another for shape classification. Although this model wasn't designed for real-time deployment, it still holds potential utility.

Sichkar and Kolyubin [14] propose a classification model for traffic sign detection together with carefully chosen evaluation metrics and baseline results. In their evaluation, they separated sign detection from classification and also measured the performance of relevant categories of signs to allow for benchmarking specialized solutions. Further, they use different filters of CNN using the GTSDB dataset.

Faster RCNN is an object detection architecture presented by Ross Girshick, Shaoqing Ren, Kaiming He, and Jian Sun in 2015 and is one of the famous object detection architectures that use convolution neural networks like YOLO (You Look Only Once) and SSD (Single Shot Detector).

Gao et al. [5] proposed a study that improves traffic sign detection for autonomous vehicles by addressing limitations in Faster R-CNN. It integrates feature fusion for weather robustness, deformable convolution for improved sign recognition, and ROI Align for accurate distant sign detection. This method outperforms existing methods in challenging environments.

RetinaNet, introduced by Lin et al. [10], is a single-stage object detection architecture known for its high accuracy, particularly in detecting small objects. It employs a focal loss to address class imbalance and uses Feature Pyramid Networks (FPN) to handle objects at multiple scales. RetinaNet has been effectively applied in various traffic sign detection tasks due to its robustness and performance.

Ellahyani, Jaafari, and Charfi [4] employed color, shape, and machine learning algorithms to detect and recognize traffic signs. This research contributes to intelligent transportation systems designed to alert drivers about safety measures and sign-related details. Additionally, the study offers comparative insights into these methods.

However, it is noteworthy that in the above-mentioned works, either the authors use CNN for traffic sign recognition or OpenCV for traffic sign detection and extraction. Our paper proposes a custom TS-Net and TSDetector CNN model to classify and detect the traffic sign and test this model using the images extracted from real-time using PyTorch. To enhance the model's performance, the inclusion of these additional changes, along with improvements in their data-set sizes, holds the potential for significant improvement.

3. Data Collection

The experimentation utilized the German Traffic Sign Detection Benchmark (GTSDB), introduced at the IEEE International Joint Conference on Neural Networks 2013 [6]. The dataset is publicly available at https://benchmark.ini.rub.de/gtsdb_dataset.html. It comprises 900 images containing 1206 traffic signs, divided into a training set of 600 images and a testing set of 300 images, stored in Portable Pixmap (ppm) format. The signs vary in size from 16x16 to 128x128 pixels and may appear singly or in multiples, with variations in orientation, lighting, and occlusions.

The signs are categorized into four types: mandatory, prohibitory, danger, and other. However, signs labeled as "other" are irrelevant to the challenge and thus excluded. The training set contains 396 prohibitory (59.5%), 114 mandatory (17.1%), and 156 danger (23.4%) sign samples, while the testing set comprises 161 prohibitory, 49 mandatory, and 63 danger sign images.

The dataset encompasses 43 different traffic sign classes, commonly found worldwide. Dataset statistics are depicted in Table 1.

3.1. Statistics of Our GTSDB Dataset

Table 1. Statistics of our GTSDB dataset

Data Split	Training	Test	Total
Numbers	600	300	900

3.2. Traffic Sign Classification Labels

Table 2. Traffic Sign Classification Labels (Part 1)

ID	Description	Type
0	speed limit 20	prohibitory
1	speed limit 30	prohibitory
2	speed limit 50	prohibitory
3	speed limit 60	prohibitory
4	speed limit 70	prohibitory
5	speed limit 80	prohibitory
6	restriction ends 80	other
7	speed limit 100	prohibitory
8	speed limit 120	prohibitory
9	no overtaking	prohibitory
10	no overtaking (trucks)	prohibitory
11	priority at next intersection	danger
12	priority road	other
13	give way	other
14	stop	other
15	no traffic both ways	prohibitory
16	no trucks	prohibitory
17	no entry	other
18	danger	danger
19	bend left	danger
20	bend right	danger

Figure 1. Traffic sign images of all the 43 classes.

Table 3. Traffic Sign Classification Labels (Part 2)

ID	Description	Type
21	bend	danger
22	uneven road	danger
23	slippery road	danger
24	road narrows	danger
25	construction	danger
26	traffic signal	danger
27	pedestrian crossing	danger
28	school crossing	danger
29	cycles crossing	danger
30	snow	danger
31	animals	danger
32	restriction ends	other
33	go right	mandatory
34	go left	mandatory
35	go straight	mandatory
36	go right or straight	mandatory
37	go left or straight	mandatory
38	keep right	mandatory
39	keep left	mandatory
40	roundabout	mandatory
41	restriction ends (overtaking)	other
42	restriction ends (overtaking (trucks))	other

4. Methodology

4.1. Data Loading and Pre-Processing

Our study initiates with the procedure of loading and preprocessing the data essential for both the training and testing phases. Instead of utilizing the PyTorch `datasets.ImageFolder` class, we implement a custom dataset class named `GermanTrafficSignDataset` to handle our specific dataset structure. The dataset comprises images segregated into subdirectories by class labels, representing different types of traffic symbols.

For both training and testing datasets, we apply a consistent set of transformations to ensure uniformity in the data input to our model. These transformations include resizing the images to 32x32 pixels, converting the images to tensor format, and normalizing them with mean and standard deviation values of (0.5, 0.5, 0.5) for each channel, respectively. Notably, we do not perform any data augmentation techniques such as random horizontal flips or rotations on the training dataset; instead, we maintain a uniform preprocessing strategy for both datasets to preserve the original orientation and structure of the traffic signs, which are critical for accurate classification.

Following the pre-processing, we partition the dataset into training and testing sets, constituting 80% and 20% of

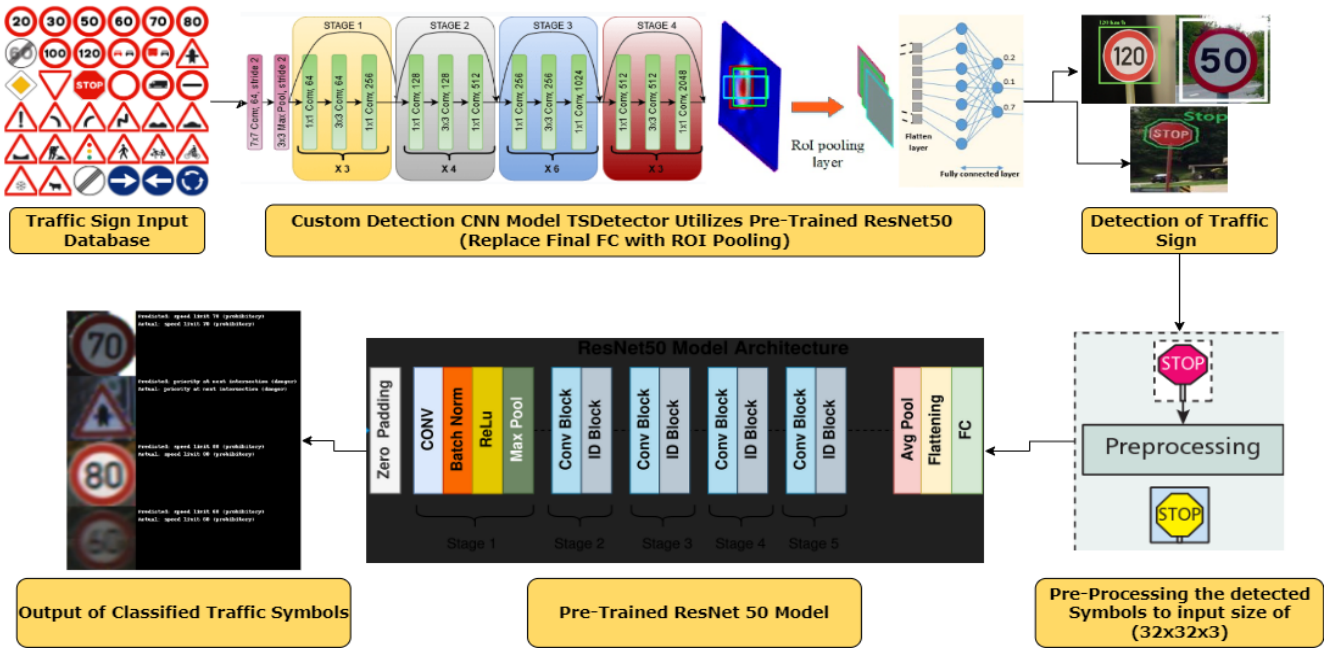

Figure 2. Architecture Diagram of the Project.

the dataset, respectively. We then instantiate DataLoader objects, `train_loader` and `test_loader`, with a batch size of 64. The training data loader shuffles the data to introduce randomness, aiding in better generalization during the model training process.

4.2. Data Transformation Strategies

Our data transformation strategy is fundamental in standardizing the dataset for optimal processing by the deep learning model. Contrary to typical augmentation practices aimed at introducing variability within the training dataset, our approach maintains uniformity across both training and testing datasets. This strategy involves resizing all images to a fixed dimension of 32x32 pixels and normalizing them. The normalization process adjusts pixel values across all channels with mean and standard deviation values set to (0.5, 0.5, 0.5), thereby standardizing the images for consistent model input.

This uniform approach to data transformation, devoid of augmentation such as random flips or rotations, is deliberate. Given the nature of our dataset, which comprises various traffic signs, preserving the original orientation and appearance of the signs is crucial. Traffic signs are designed to convey specific information based on their shape, color, and symbols; thus, altering their orientation or appearance could potentially mislead the model, impacting its ability to learn accurate and generalizable features. Therefore, our transformations focus solely on resizing and normalizing the images, ensuring that the integrity and specific characteristics of each traffic sign are retained for effective model training and evaluation.

4.3. Defining TSDetector

The Custom Traffic Signal Object Detection CNN model - TSDetector, central to our study, is specifically designed for the task of traffic sign detection. The structure of the model, comprising a sequence of layers, is detailed below:

- **Backbone (ResNet50)**: Incorporates a pre-trained ResNet50 model as its backbone. This backbone serves as a feature extractor for the model. The final fully connected layer of the ResNet50 model is discarded, retaining only the preceding layers. This adjustment allows the model to focus on extracting meaningful features from input images, utilizing the capabilities of ResNet50 to identify intricate patterns and structures within the data.
- **RPN (Region Proposal Network)**: The Region Proposal Network generates region proposals (potential object bounding boxes) that are further processed by the RoI pooling layer.
- **Pooling Layers (RoI Pooling)**: RoI (Region of Interest) pooling is performed in the forward pass. RoI pooling allows for the extraction of features from specific regions proposed by the model, enabling focused analysis and classification of these regions. By deferring pooling until the forward pass, the model retains flexibility in handling variable-sized regions of interest, optimizing its performance for object detection tasks.

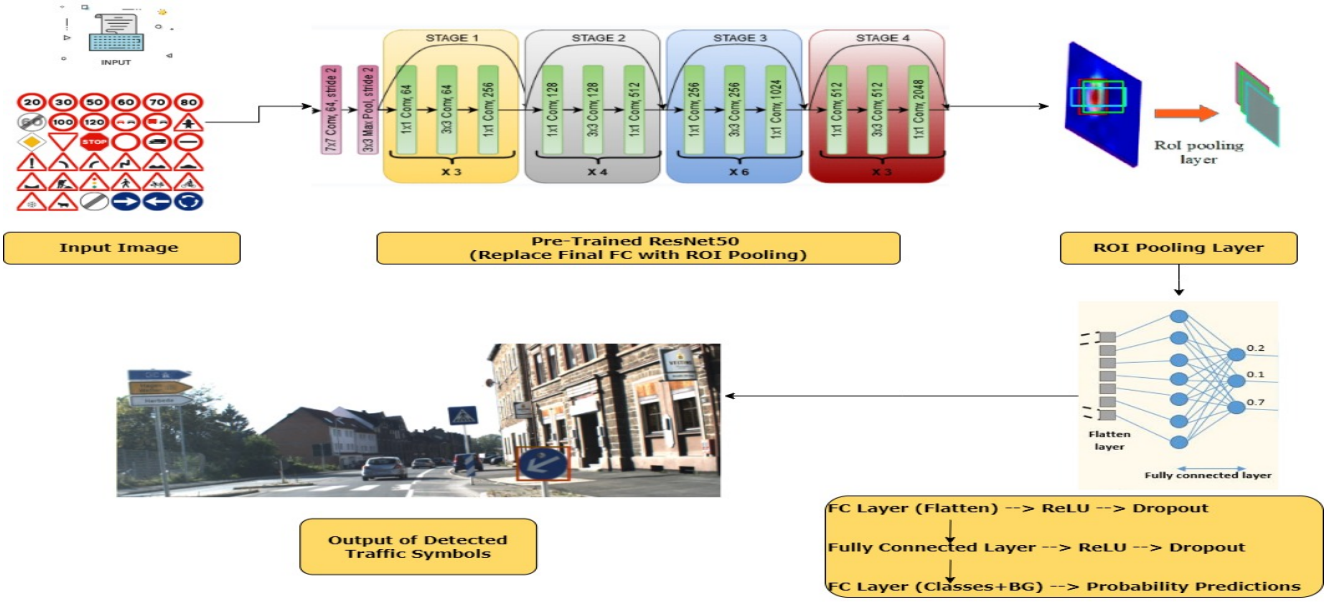

Figure 3. Architecture Diagram of TSDetector (The Custom Traffic Signal Object Detection Model)

- **Fully Connected Layer**: In the fully connected layer stage, the backbone features undergo a sequence of transformations. Initially, these features are flattened into a vector representation. Following this, two consecutive fully connected layers are applied, each incorporating Rectified Linear Unit (ReLU) activation and dropout regularization. The final layer of this sequence is a fully connected layer, where the number of neurons corresponds to the total number of classes plus an additional one for the background class. This layer's output serves as the classifier's prediction, providing probabilities for each class, including the background class.

4.4. Defining TSNet

The The Custom Traffic Symbol Classification Model - TSNet, central to our study, is specifically designed for the task of traffic sign classification. Its architecture, depicted in Figure 4, simplifies the classification challenge by focusing on efficient processing and identifying key features within the images. The structure of the model, comprising a sequence of layers, is detailed below:

- **Layer 1 (Convolutional)**: This initial layer is a convolutional layer equipped with 32 filters, a kernel size of 3x3, stride of 1, and a padding of 1. It is followed by batch normalization and ReLU activation, aiming to capture basic visual features from the input images.
- **Layer 2 (Convolutional)**: The second layer, another convolutional layer, increases the depth with 64 filters. Similar to the first layer, it uses a kernel size of 3x3, stride of 1, and padding of 1, followed by batch normalization and ReLU activation. This layer further refines the feature extraction process.
- **Layer 3 (Convolutional)**: Continuing the pattern, this layer employs 128 filters for even deeper feature extraction. With the same kernel size, stride, and padding as the previous layers, followed by batch normalization and ReLU activation, it enhances the model's ability to recognize more complex patterns.
- **Pooling Layers**: After each convolutional layer, a max-pooling layer with a 2x2 filter and stride of 2 is applied. These layers serve to reduce the spatial dimensions of the feature maps, effectively compressing the extracted features and reducing computation in subsequent layers.
- **Global Average Pooling**: Following the convolutional stages, a global average pooling layer condenses each feature map to a single value, focusing the model on the most essential information.
- **Fully Connected Layer**: The model concludes with a fully connected layer that transforms the pooled features into final class scores, corresponding to the 43 classes of traffic signs.

4.5. Training Details

In the training phase of our models, we adopt a batch size of 64, aiming to balance between computational efficiency

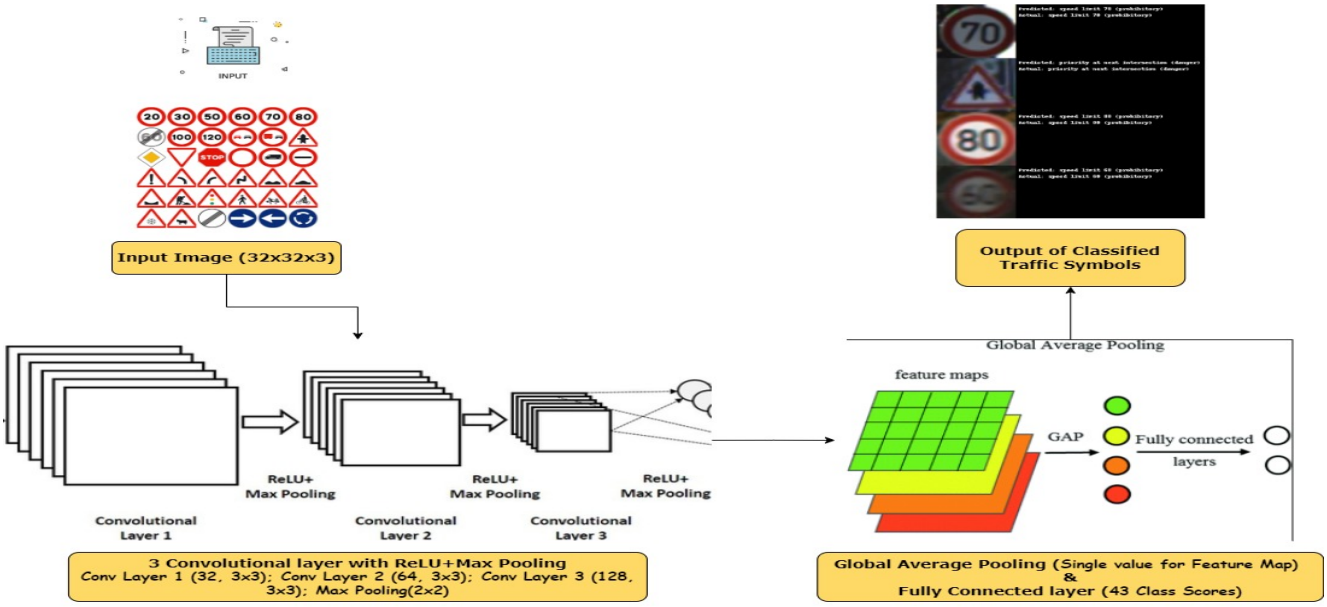

Figure 4. Architecture Diagram of TSNet (The Custom Traffic Symbol Image Classification Model)

and the ability to generalize across the dataset. The training spans 25 epochs for the pre-trained models (ResNet50 and MobileNetV2) and 100 epochs for the custom models (TSNet and TSDetector). This duration is determined to allow sufficient learning while preventing potential overfitting on the training data.

We employ the Adam optimizer [8], renowned for its effectiveness in various deep-learning tasks, with an initial learning rate set to 0.001. This choice is motivated by Adam's adaptive learning rate properties, which help in navigating the complex landscape of neural network parameter optimization.

The entire training regimen is conducted within a GPU-enabled environment, leveraging the accelerated computing resources to significantly reduce the time required for each training epoch. This setup is crucial for handling the computational demands of processing high volumes of image data and executing the backpropagation algorithm essential for neural network training.

During training, our model iteratively processes batches of images supplied by the `train_loader`, each batch comprising randomly selected samples to ensure diverse exposure to the dataset's characteristics. For each batch, the model computes the cross-entropy loss — a standard choice for multi-class classification problems — between its predictions and the true labels. This loss quantifies the model's performance, guiding the optimizer in adjusting the model parameters to minimize the loss, thereby improving the model's accuracy.

To ensure the model's generalization capabilities and monitor its performance on data it has not encountered during training, we validate the model at the end of each epoch using the `test_loader`. This loader provides a separate dataset not involved in the training process, allowing us to evaluate the model's accuracy and other relevant metrics on unseen data. This practice is instrumental in detecting and mitigating overfitting, ensuring the model's efficacy extends beyond the training dataset to accurately classify new, real-world images of traffic signs.

4.6. Implementation Details

Our implementation is carried out using the PyTorch framework [12], which offers the flexibility and efficiency required for this kind of deep learning task. The code and datasets used in this study are made available for public access, encouraging reproducibility and further research in this domain.

4.7. Evaluation of Model-Detection Metrics

For evaluating object detection models like TSDetector, we used Intersection over Union (IoU), precision, recall, and mean Average Precision (mAP).

4.7.1 Intersection over Union (IoU)

IoU measures the overlap between the predicted and ground-truth bounding boxes. The formula is:

$$\text{IoU} \quad \frac{\text{Area of Intersection}}{\text{Area of Union}} \tag{1}$$

An IoU score of 0.5 or higher is generally considered as a correct detection.

4.7.2 Precision and Recall

Precision measures how many predicted bounding boxes were actually relevant, while recall measures how many of the relevant bounding boxes were successfully retrieved. The formulas are:

$$\text{Precision} \quad \frac{TP}{TP + FP} \tag{2}$$

$$\text{Recall} \quad \frac{TP}{TP + FN} \tag{3}$$

where: - TP (True Positive): Correct detections. - FP (False Positive): Incorrect detections. - FN (False Negative): Missed detections.

4.7.3 Mean Average Precision (mAP)

The mean Average Precision is the mean of the average precision values across all classes. Average precision is calculated using the precision-recall curve:

$$\text{mAP} \quad \frac{1}{n} \sum_{i=1}^{n} \text{AP}_i \tag{4}$$

where n is the number of classes and AP_i is the average precision for class i.

4.8. Evaluation of Model-Classification Metrics

Assessing the model's effectiveness and efficiency involves crucial procedures in performance measurement and evaluation. Our classification model employs various key performance metrics, including accuracy, precision, recall, and F1-score. Accuracy quantifies the ratio of correctly classified results to the total number of outcomes. Precision and recall both gauge correctly predicted positives, with precision emphasizing the proportion of accurately placed positive weights, while recall indicates how many positive weights the model successfully detected. The F1 score represents the harmonic mean of precision and recall. These classification metrics can be operationalized through a confusion matrix to assess the performance of each model in the classification of traffic signs. The mathematical expressions given in Equations 5 to 8 are used to calculate the considered metrics.

$$\text{Precision} \quad \frac{TP}{TP + FP} \tag{5}$$

$$\text{Recall} \quad \frac{TP}{TP + FN} \tag{6}$$

$$\text{F1-score} \quad \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7}$$

$$\text{Accuracy} \quad \frac{TP + TN}{TP + FP + FN + TN} \tag{8}$$

Here, TP, FP, TN, and FN represent true positive, false positive, true negative, and false negative, respectively.

5. Results

This section presents the performance outcomes of various deep learning models employed in our study. We analyze these models based on their training and validation metrics, along with their test dataset performance depicted through confusion matrices and classification reports.

5.1. Object Detection Results

5.1.1 TSDetector Results

The custom TSDetector model was trained for 100 epochs using the GTSDB dataset. The results are as follows:

Table 4. Performance Metrics for TSDetector

Metric	TSDetector
Precision	0.47
Recall	0.47
F1 Score	0.47
Average IoU	0.26

5.1.2 Comparison with Pre-trained Models

We compare the custom TSDetector model against the pre-trained Faster RCNN and RetinaNet models, trained for 25 epochs on the same GTSDB dataset.

Table 5. Comparison of Model Performance on the Test Dataset (Object Detection)

Metric	Faster RCNN	RetinaNet	TSDetector
Precision	0.44	0.42	0.47
Recall	0.44	0.42	0.47
F1 Score	0.44	0.42	0.47
Average IoU	0.11	0.01	0.26

The TSDetector model outperformed both Faster RCNN and RetinaNet in terms of precision, recall, and Average IoU.

5.2. Classification Results

5.2.1 TSNet Results

The custom TSNet model was trained for 100 epochs using the GTSDB dataset. Its results are presented below:

Table 6. Performance Metrics for TSNet

Metric	TSNet
Precision	0.743
Recall	0.663
F1 Score	0.645
Accuracy	81.07%

5.2.2 Comparison of Classification Models

We compare the custom TSNet model against pre-trained ResNet50 and MobileNetV2 models.

Table 7. Comparison of Classification Model Performance

Metric	ResNet50	MobileNetV2	TSNet
Precision	0.973	0.957	0.743
Recall	0.971	0.957	0.663
F1 Score	0.969	0.957	0.645
Accuracy	97.12%	95.88%	81.07%

5.3. Integrated Output (TSDetector + ResNet50)

To achieve an end-to-end traffic sign detection and classification solution, we integrated the TSDetector model for detection with the ResNet50 model for classification. The output is shown in Figure 5.

6. Discussion of Results

The final results for the TSNet model reveal a precision score of 0.743, a recall score of 0.663, an F1 score of 0.645, and an accuracy of 81.07%. Although TSNet's performance metrics are lower than those of established models like ResNet50 and MobileNetV2 in terms of precision, recall, F1 score, and accuracy, its ability to identify specific classes with reasonable precision suggests potential. However, its lower recall and F1 scores indicate a need for improvement in recognizing a broader range of traffic sign types.

On the other hand, the custom object detection model, TSDetector, demonstrates a precision score, recall score, and F1 score of 0.47. In comparison, Faster RCNN and RetinaNet models achieve slightly lower metrics of 0.44 and 0.42, respectively. TSDetector also outperformed the pre-trained models in terms of Average IoU, with a score of 0.26 compared to 0.11 (Faster RCNN) and 0.01 (RetinaNet). Despite slightly lower precision and recall, TSDetector excels in accurately localizing objects, evident from its higher Average IoU. This indicates promising performance in object detection tasks, particularly in accurately delineating object boundaries.

The integrated TSDetector + ResNet50 model shows promising potential for end-to-end traffic sign detection and classification tasks. While current metrics indicate potential for further refinement and optimization, their application in traffic sign classification and detection tasks offers valuable insights for future research and development in this field.

The results also highlight some key takeaways:

- **Model Improvements**: Both TSNet and TSDetector exhibit potential for improvement through data augmentation, hyperparameter tuning, and training on more extensive datasets.
- **Pre-trained Model Comparison**: Pre-trained models such as ResNet50 and MobileNetV2 maintain high accuracy, precision, and recall scores, providing reliable benchmarks for custom models.
- **Integrated Solutions**: The integration of object detection and classification models (TSDetector + ResNet50) offers a comprehensive traffic sign recognition solution, vital for autonomous vehicle systems.
- **Generalization**: Custom models like TSNet and TSDetector can provide robust solutions for specific tasks if carefully tuned and validated.

This comprehensive evaluation provides valuable insights into the current state and future direction of traffic sign recognition technology, paving the way for the development of more accurate and efficient models tailored to real-world traffic sign interpretation.

7. Future Direction

The results of this study underscore the significance of model selection in tasks related to traffic sign recognition and detection. Custom models such as TSNet and TSDetector introduce a novel perspective, highlighting the potential contributions of tailor-made models to this domain. Our comparison elucidates pivotal aspects of model architecture and training that influence effective classification performance, offering guidance for future endeavors in model development for real-time traffic sign recognition and detection.

Moving forward, our research can explore several promising directions:

- **Data Augmentation and Balance**: Enhancing data augmentation strategies and addressing class imbalances may improve TSNet and TSDetector's generalization capabilities.
- **Ensemble Models**: Implementing an ensemble approach with pre-trained models like ResNet50 and MobileNetV2 could increase overall classification performance.

Figure 5. Integrated output of TSDetector + ResNet50 models on GTSDB dataset.

- **Attention Mechanisms**: Incorporating attention mechanisms, such as the self-attention module or the spatial attention module, could help refine detection and classification accuracy.
- **Real-World Scenarios**: Testing the models in real-world scenarios using different camera angles, weather conditions, and lighting could further validate their robustness.
- **Lightweight Models**: Developing lightweight versions of TSNet and TSDetector optimized for deployment on edge devices will improve the practical application of our models in real-time traffic sign detection and classification.

Through this comparison and exploration of TSNet and TSDetector, we lay the foundation for future advancements in traffic sign recognition technology. By addressing current limitations and leveraging the insights gained from this comparative analysis, our forthcoming work can pave the path toward the development of more precise, efficient, and resilient models specifically tailored to the complexities of real-world traffic sign interpretation.

8. Conclusion

This study explored the application of deep learning in the classification and detection of traffic signs, providing a comprehensive evaluation of a custom-designed CNN model named TSNet and TSDetector alongside established pre-trained models.

In conclusion, the evaluation of the TSNet and TSDetector models in traffic sign recognition and detection tasks provides valuable insights into their performance and potential contributions to the field. While TSNet exhibits slightly lower performance metrics compared to established models like ResNet50 and MobileNetV2, its ability to identify specific classes with reasonable precision highlights its potential utility. However, its lower recall and F1 scores suggest a need for enhancement in recognizing a broader range of traffic sign types.

Conversely, TSDetector demonstrates higher precision, recall, and F1 scores compared to Faster RCNN and RetinaNet models, highlighting its ability to accurately delineate object boundaries. With an Average IoU of 0.26, TSDetector excels in accurately localizing traffic signs, indicating promising performance in object detection tasks.

Both TSNet and TSDetector can undergo further refinement and optimization through iterative development and fine-tuning processes. Despite their current metrics being lower compared to established models, their potential for improvement positions them as valuable foundations for future research and development efforts in traffic sign classification and detection tasks. TSNet, in particular, introduces a new dimension, emphasizing the potential contributions of custom-designed models to the field, and laying the groundwork for further advancements in traffic sign recognition technology.

References

[1] H. Akatsuka and S. Imai. Road signposts recognition system. *Proceedings of the IFAC Workshop on Vision, Recognition, Action Control*, pages 6–9, 1987. 2

[2] Abdelfettah Amira, Salah Hamdi, Ahmed Wali, and Abdenour Hadid. Olimp: a multimodal dataset for object detection in a natural environment. In *2020 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 399–404. IEEE, 2020. 1

[3] Bilel Benjdira, Oussama Hegazy, Abdelouahab Ahmad, and Zakaria Moutakki. A comprehensive study of traffic sign detection and recognition: The case of vision-based systems. *Journal of Imaging*, 7:211, 2021. 1

[4] A. Ellahyani, I. E. Jaafari, and S. Charfi. Traffic sign detection and recognition based on color, shape, and machine learning algorithms. *Proceedings of the 2022 IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1458–1465, 2022. 3

[5] Jiayuan Gao, Wei Zhao, Xiao Zhang, and Hui Gao. Improving traffic sign detection for autonomous vehicles. *Electronics*, 11:1488, 2022. 3

[6] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. Detection of traffic signs in real-world images: The german traffic sign detection benchmark. In *International Joint Conference on Neural Networks (submitted)*, 2013. 1, 2, 3

[7] Niaz Islam. Traffic sign classification using convolutional neural network (cnn). *Proceedings of the 2019 10th International Conference on Information and Communication Systems (ICICS)*, pages 53–58, 2019. 3

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2014. 7

[9] Hee Seok Lee and Wooil Kim. Simultaneous traffic sign detection and boundary estimation using convolutional neural network. *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3028–3036, 2018. 2

[10] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017. 3

[11] Pranjali Pandey, Vimal Kumar, and A. Tripathi. A hybrid approach for detection and recognition of traffic signs. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 7:309–322, 2014. 2

[12] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, pages 8024–8035, 2019. 7

[13] F. Shangzheng. A fast traffic sign recognition model based on improved convolutional neural network. *Proceedings of the 2018 Chinese Automation Congress (CAC)*, pages 2505–2509, 2018. 2

[14] Valentyn Sichkar and Sergey A. Kolyubin. Effect of various dimension convolutional layer filters on traffic sign classification accuracy. *Proceedings of the 2018 IEEE International Conference on Artificial Intelligence and Soft Computing (ICAISC)*, pages 361–370, 2018. 3

[15] Kristy Tsoi and Aubrey Wheelus. A cost-sensitive deep learning system for traffic sign recognition. *Proceedings of the 2020 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 361–365, 2020. 2