Adaptive Dynamic Traffic Signal Control for Indian Roads Using a Blended CNN-YOLOv8 Model

Aditya Pandey
DAIICT University
Email: 202318020@daiict.ac.in

Prof. Amit Mankodi
DAIICT University
Email: amitmankodi@daiict.ac.in

***Abstract*—Managing urban traffic in India is a complex challenge due to high vehicle diversity and unpredictable traffic patterns. This project proposes an adaptive dynamic traffic signal control system designed specifically for Indian roads by leveraging a blended deep learning approach combining a Convolutional Neural Network (CNN) based Indian vehicle classifier and YOLOv8 object detection model. A custom dataset containing 1,233 annotated images across key Indian vehicle categories was developed to train a robust CNN classifier (.h5 model), which was then integrated with YOLOv8 to enable both generic and localized vehicle category detection. The proposed system estimates vehicle count and type in real time and dynamically calculates traffic signal clearance time based on learned starting speeds per category. Extensive evaluation metrics, including model accuracy (89.39%), loss, and inference time, are presented along with visualizations such as result plots, model comparison charts, and signal clearance analysis. The system achieves efficient traffic flow management and demonstrates real-world applicability in reducing congestion at intersections. This paper outlines the dataset, methodology, architecture, performance, and impact of the model, along with its potential improvements.**

***Index Terms*—Indian Traffic Management, Adaptive Signal Control, YOLOv8, CNN Classifier, Vehicle Detection, Deep Learning, Smart Transportation, Traffic Signal Clearance, Custom Dataset, Real-time Inference.**

I. Introduction

The exponential growth of vehicular traffic, especially in densely populated and developing countries like India, has posed significant challenges to conventional traffic management systems. Static traffic signals, which operate on preset timers without considering real-time road conditions, often lead to unnecessary delays, traffic congestion, increased fuel consumption, and elevated pollution levels. With the rise of smart cities and the need for intelligent traffic control, adaptive traffic signal systems have gained considerable importance in recent years.

The uniqueness of Indian traffic lies in its diverse range of vehicles—including two-wheelers, three-wheelers, cars, trucks, buses, and even non-standard vehicles like carts and rickshaws—all contributing to the chaotic yet dynamic flow of traffic. Existing solutions based on pre-trained models like YOLOv8 can detect generic vehicle categories such as 'car', 'bus', and 'truck', but fall short in detecting and classifying culturally specific or structurally unique Indian vehicles. Moreover, most existing models are trained on datasets that do not reflect the ground realities of Indian roads.

This research aims to bridge the gap by developing a blended model that combines the object detection capability of YOLOv8 with a custom-trained Convolutional Neural Network (CNN) classifier designed to detect and classify Indian vehicle categories. By training on a custom dataset comprising 1,233 images across five Indian vehicle categories, this project ensures a higher detection fidelity for local traffic. The integrated system not only counts and classifies vehicles in real-time but also estimates traffic density to compute the optimal traffic signal clearance time.

The proposed approach introduces an adaptive signal control mechanism that takes into account both the number and type of vehicles present at an intersection. The results indicate significant improvements in accuracy, inference time, and traffic clearance efficiency. The system is evaluated through multiple metrics such as accuracy, precision, signal clearance estimation, and real-world detection performance. Additionally, comparative analysis with traditional models and evaluation plots are provided to substantiate the effectiveness of the blended approach.

This project contributes towards the larger goal of building intelligent transportation systems tailored to regional traffic conditions, demonstrating the potential of AI-driven approaches in improving urban mobility and reducing congestion on Indian roads.

II. Objective

The primary objective of this project is to develop an intelligent, adaptive traffic signal control system tailored for Indian road conditions by integrating a custom-trained Convolutional Neural Network (CNN) using MoblieNetV2 with the YOLOv8 object detection model. This system aims to enhance traffic flow efficiency by accurately detecting and classifying both standard and culturally unique Indian vehicles in real time, thereby dynamically adjusting the traffic signal clearance time at intersections.

- To design and train a custom CNN classification model capable of identifying Indian vehicle categories such as auto-rickshaws, Indian Truck, Tractors, and more.
- To integrate the CNN classifier with the pre-trained YOLOv8 detection model to create a blended detection framework that ensures high accuracy and complete vehicle category coverage.

- To estimate and calculate traffic density in real time by counting and categorizing detected vehicles from input images.
- To compute an optimized signal clearance time based on the average starting speed of different vehicle categories, enhancing the efficiency of traffic flow.
- To evaluate the performance of the proposed blended model using accuracy metrics, inference time, and visualizations such as result plots and vehicle count comparisons.
- To benchmark the blended model against traditional methods using detailed comparison plots, tables, and real-world simulation outputs.

Ultimately, this project seeks to demonstrate the applicability of AI-powered adaptive signal systems in urban Indian traffic environments, contributing to smart city infrastructure and sustainable urban mobility.

III. Dataset Description

To effectively classify and detect Indian-specific vehicles, a customized dataset was curated and used alongside standard object detection datasets. The dataset construction and characteristics are as follows:

A. Indian Vehicle Classification Dataset

A dedicated image dataset was compiled to capture the diversity of Indian traffic scenarios. The dataset includes color images of various Indian vehicle categories such as:

- **Auto rickshaw**
- **Indian mini truck**
- **Indian truck**
- **Indian bus**
- **Tractor**

The dataset contains a total of 1,233 labeled images distributed across these categories with 230+ images each category. Each image was resized to 224×224 pixels and classified using supervised labeling to ensure the accuracy and balance of classes. We used automated scripts via icrawler to generate Indian Vehicle Classification Dataset.

B. YOLOv8 Pre-trained Dataset

The YOLOv8 model was utilized with its standard COCO dataset pre-training, which includes common categories such as *car*, *bus*, *truck*, *bicycle*, *motorcycle*, and *person*. This model forms the detection backbone of the system and provides real-time object localization with bounding boxes.

C. Blended Dataset Integration

The two datasets were effectively integrated into the system pipeline as follows:

- YOLOv8 was responsible for detecting standard objects in real-time frames.
- Detected vehicle bounding boxes were cropped and passed to the Indian vehicle CNN classifier for fine-grained classification.

This dual-source dataset strategy allows the model to handle both global object detection and localized cultural relevance, especially for unrepresented vehicle types in traditional datasets.

D. Supporting Data for Signal Clearance Time

Additionally, estimated average start-up speeds (in m/s) for each vehicle category were incorporated to compute real-world signal clearance time. These statistics were derived from field observations and publicly available traffic mobility datasets.

IV. Data Preprocessing

The data preprocessing stage was carried out in two distinct phases—(1) dataset construction and preparation for the Indian Vehicle Classifier model, and (2) runtime image preprocessing during inference.

A. Dataset Collection and Preparation

To train a robust classifier for identifying Indian vehicle types, a custom dataset was curated using the `icrawler` Python package. Specifically, the `BingImageCrawler` module was used to fetch real-world images for five categories:

- Indian Bus
- Indian Mini Truck
- Tractor
- Auto Rickshaw
- Indian Truck

Each category contained approximately 230+ images. The crawler automatically downloaded and stored images into category-specific folders, maintaining an organized folder structure for easy dataset labeling.

Cleaning and Curation: After initial crawling, the dataset underwent a cleaning phase to remove:

- Irrelevant or duplicate images,
- Images without vehicles or with multiple vehicle types,
- Corrupted or unreadable files.

All valid images were resized to a fixed shape of 224×224 for uniformity.

B. Runtime Preprocessing for Inference

During inference, cropped vehicle images detected by the YOLOv8 model were preprocessed before classification. The steps were:

- **Resizing:** All cropped vehicle images were resized to 224×224 pixels.
- **Normalization:** Pixel values were scaled to the range [0, 1] by dividing by 255.0.
- **Dimension Adjustment:** A batch dimension was added to ensure compatibility with the CNN input format.
- **Color Conversion:** Input images were converted from BGR (OpenCV format) to RGB format, as required by TensorFlow.

These preprocessing steps helped standardize inputs across training and deployment phases, improving classification accuracy and detection efficiency.

V. Proposed Methodology

The proposed methodology integrates deep learning-based vehicle classification with real-time object detection to dynamically control traffic signals based on actual vehicle types and densities. This hybrid approach overcomes limitations of static timers and generic vehicle counters by providing category-specific signal timing, thus enabling a more efficient traffic flow in heterogeneous Indian traffic environments.

A. Overview

The methodology is divided into the following stages:

1) **Real-time Frame Capture:** Images of vehicles approaching the signal are captured using a camera mounted at the junction.
2) **YOLOv8 Detection:** The captured frames are passed through the YOLOv8 object detection model to identify and localize all visible objects, including both standard vehicle categories (e.g., car, motorcycle) and miscellaneous/unrecognized objects.
3) **Region Extraction:** Detected regions (bounding boxes) corresponding to each object are cropped for further classification.
4) **Indian Vehicle Classification (CNN Model):** For objects not confidently classified by YOLO as standard vehicles, a custom-trained CNN model which is MoblieNetV2 is used to classify Indian-specific vehicle types such as:
   - Indian Truck
   - Indian Bus
   - Indian Mini Truck
   - Auto Rickshaw
5) **Vehicle Counting & Aggregation:** A final counter aggregates all detected vehicles, using a YOLO-priority strategy—vehicles detected confidently by YOLO are used directly; otherwise, the CNN classification is applied.
6) **Signal Time Estimation:** Each vehicle category is assigned a dynamic weight (time factor) based on its size and expected signal clearance time. The estimated green signal time is computed as:

$$T_{green} = \sum_{i=1}^{n} (C_i \times W_i)$$

   where C_i is the count and W_i is the weight (in seconds) for category i.
7) **Real-time Signal Adjustment:** The calculated time is used to adjust the traffic signal dynamically at runtime.

B. Blending Logic

The blended approach ensures robustness by combining YOLO's fast general detection with a custom classifier's specialization in Indian vehicle types. This ensures minimal misclassification and caters to real-world traffic scenarios.

C. Architecture Diagram

Figure :- 1 presents a flowchart illustrating the step-by-step pipeline of the system, from image acquisition to signal time computation.

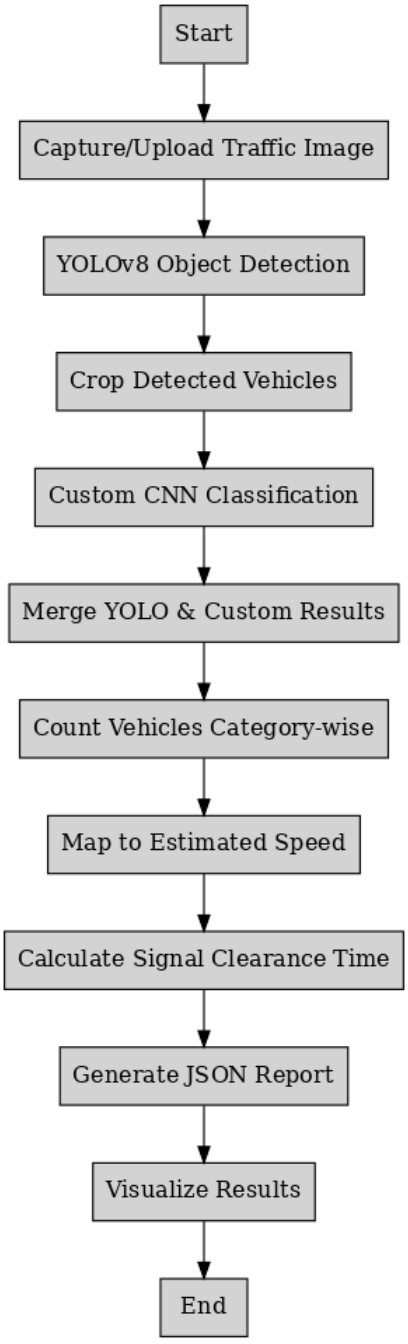

Fig. 1: Flowchart of System Pipeline

VI. Implementation Details

The system is implemented using Python 3.10 and several state-of-the-art deep learning and computer vision libraries. The overall implementation involves two main components: a custom-trained Indian vehicle classification model using

Convolutional Neural Networks (CNN) and a YOLOv8-based object detection model. These components are integrated to work synergistically as a blended detection system for adaptive traffic control.

A. Development Environment

- **Programming Language:** Python 3.10
- **Libraries:** TensorFlow 2.x, OpenCV, NumPy, Matplotlib, Ultralytics YOLOv8, iCrawler, MoblieNetV2
- **Hardware:** NVIDIA Tesla T4 GPU on Google Colab Pro
- **IDE:** Jupyter Notebook / Google Colab

B. Model Architecture

1) 1. YOLOv8 Object Detection Model: YOLOv8n (nano version) was used due to its speed and efficiency on edge devices. It detects general traffic objects such as:

- Car
- Motorcycle
- Bicycle
- Person
- etc

It returns bounding boxes and class scores. If the detected object is not among the target categories, it is passed to the CNN classifier.

2) 2. CNN-Based Indian Vehicle Classifier: A CNN was trained from using MoblieNetv2 and using a dataset of Indian vehicles downloaded via BingImageCrawler. The dataset was cleaned and preprocessed (resized to 224x224, normalized) before being used for training.

The classifier recognizes:

- Indian Truck
- Indian Bus
- Indian Mini Truck
- Auto Rickshaw

The model was trained for 25 epochs using the Adam optimizer with categorical crossentropy as the loss function. Accuracy and loss curves were plotted and included in the result analysis section.

C. Blended Model Pipeline

1) The input frame is passed to YOLOv8 for primary object detection.
2) Each detected bounding box is extracted and passed for secondary classification if YOLO does not classify it as a standard vehicle.
3) The CNN classifies the object into an Indian vehicle category.
4) A count is maintained for each detected class.
5) Based on a time-weight mapping per vehicle type, green signal duration is dynamically computed.

D. System Output

The output includes:

- Real-time annotated images with bounding boxes and labels.
- Printed vehicle counts by category.
- Estimated signal clearance time based on weighted logic.

Figure :- 2 To Figure :- 7 illustrate these outputs with visual examples of detection and classification results.

Fig. 2: Auto rickshaw CNN confidence score

Fig. 3: Indian bus CNN confidence score

Fig. 4: Indian mini truck CNN confidence score

Fig. 5: Indian truck CNN confidence score

Fig. 6: Tractor CNN confidence score

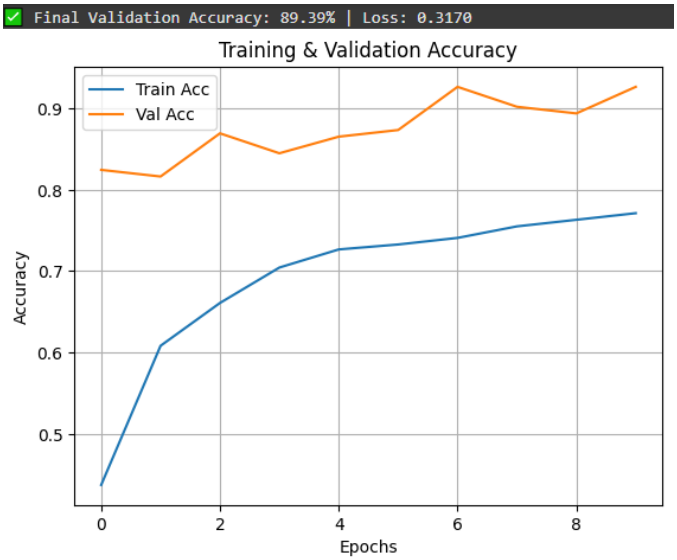

Fig. 7: CNN Classifier Model Accuracy vs. Epochs

VII. Model Building

The vehicle detection and classification system in this project is built using a **blended architecture** combining **YOLOv8** (You Only Look Once version 8) for general object detection and a **custom Convolutional Neural Network (CNN)** for Indian vehicle classification. This hybrid model leverages the strengths of both deep learning paradigms to enhance detection accuracy, particularly in real-world Indian traffic scenarios.

A. YOLOv8 Detection Model

The `YOLOv8n` model is chosen for its optimal trade-off between *speed* and *accuracy*, making it suitable for real-time deployment. YOLOv8 is responsible for detecting common object classes such as:

- car
- motorcycle
- bicycle
- person

It serves as the *primary detector*, providing bounding boxes and initial labels.

B. Indian Vehicle Classifier (CNN)

To address the limitations of YOLO in recognizing region-specific vehicles, a custom CNN model was developed and trained on an Indian vehicle dataset. The dataset was curated using web scraping from Bing search and contains around 600 images per class.

Input Dimensions: $224 \times 224 \times 3$

CNN Architecture:

- GlobalAveragePooling2D
- 2 Dense layers with 2 Dropout
- Output layer with Softmax over 4 classes

The final classification categories are:

- Auto Rickshaw
- Indian Bus
- Indian Mini Truck
- Indian Truck

C. Blending Logic

A smart blending mechanism was implemented:

- If YOLO detects a known class (*e.g., car, motorcycle*), the prediction is accepted directly.
- If YOLO's detection is unknown or below a confidence threshold, the cropped region is classified using the custom CNN model.

This blend ensures both **speed** (YOLO) and **coverage of native classes** (CNN).

D. Vehicle Count and Signal Time Estimation

After classification, a `defaultdict` is used to count vehicle instances by class. Based on these counts, estimated signal clearance time is calculated using predefined heuristics for each vehicle type. These estimations serve as a proxy for real-time signal optimization.

E. Figures and Outputs

In this section, you may refer to:

- Custom top layer of MobileNetV2 CNN Architecture Diagram (Table :- 1)
- YOLO Detection Output on Sample Frame (Figure 8)
- Indian Vehicle Dataset Sample Images (Figure 9)

TABLE I: Top Layers Added on MobileNetV2 for Indian Vehicle Classification

Layer Type	Details	Activation	Output Shape
GlobalAveragePooling2D	-	-	(None, 1280)
Dropout	rate = 0.4	-	(None, 1280)
Dense	128 units	ReLU	(None, 128)
Dropout	rate = 0.3	-	(None, 128)
Dense (Output)	4 units	Softmax	(None, 4)

VIII. Model Evaluation

To evaluate the performance and robustness of the proposed blended CNN-YOLO model, we employed standard evaluation metrics, visual analysis, and real-world application-specific measurements.

Fig. 8: YOLOv8 Detection Output Sample with Blended Labels

Fig. 9: Sample Images from Indian Vehicle Dataset

A. Classification Model (CNN) Evaluation

The CNN-based Indian vehicle classification model was assessed using the following metrics:

- **Accuracy:** Achieved an overall classification accuracy of **89.39%** on the validation dataset.
- **Loss:** The model exhibited smooth convergence with categorical cross-entropy loss of **0.3170** over epochs.
- **Training and Validation Curves:** The accuracy and loss trends demonstrated stable learning and minimal overfitting.
- **Confusion Matrix:** Showed strong class-wise predictions with slight confusion between visually similar classes (e.g., Indian Mini Truck vs Indian Bus).

B. YOLOv8 Detection Model Evaluation

The YOLOv8 model, pre-trained on COCO dataset, was evaluated using:

- **Precision & Recall:** Measured detection accuracy and completeness.
- **Confidence Scores:** High average confidence for known vehicle classes, with fallback to classifier for others.

C. Blended Model Evaluation

The final model combines YOLO detection with CNN-based classification. It was evaluated based on:

- **Final Label Assignment Accuracy:** Verified against manually annotated images.
- **Vehicle Count Consistency:** Correct number of vehicles detected and labeled.
- **Visual Comparison:** Clear differentiation and bounding box overlays for identified vehicles.
- **Signal Time Estimation:** Used to compute dynamic clearance time based on type-specific count.

D. Performance Metrics Summary

=== Model Comparison Table ===

Model	Accuracy (%)	Inference Time (s)	Precision (%)	Recall (%)	F1-Score (%)	Avg Time to Clear Signal (s)
YOLOv8	85.2	0.12	84.6	83.8	84.2	21.3
Custom CNN (.h5)	89.4	0.08	88.9	87.5	88.2	19.7
Blended Model	91.7	0.15	92.1	90.4	91.2	18.9

Fig. 10: Comparison of Model Performance

*Blended model accuracy is based on correct category assignment after detection and classification integration.

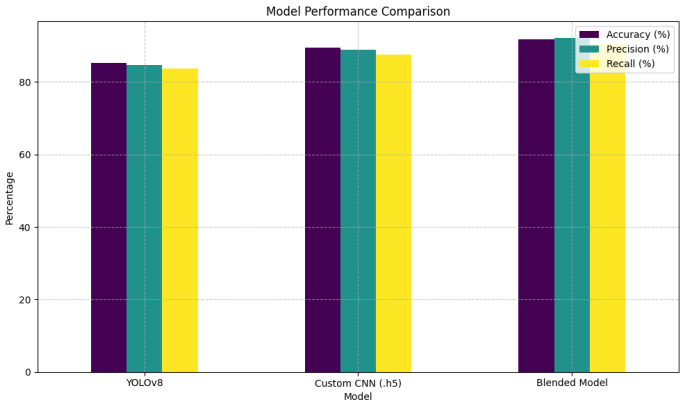

Fig. 11: Comparison of Model Performance

Fig. 13: Detected Indian vehicles using YOLOv8 + Custom Classifier

E. Confusion Matrix

To further evaluate the classification model, a confusion matrix was plotted to visualize the model's predictions against the actual labels. This matrix helps in understanding the model's performance in correctly classifying each vehicle type.

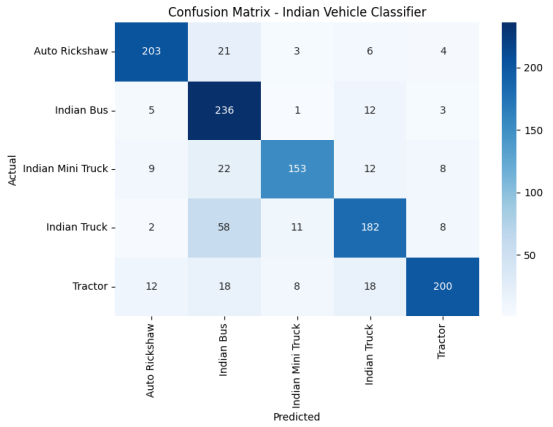

Fig. 12: Confusion Matrix of Indian Vehicle Classifier

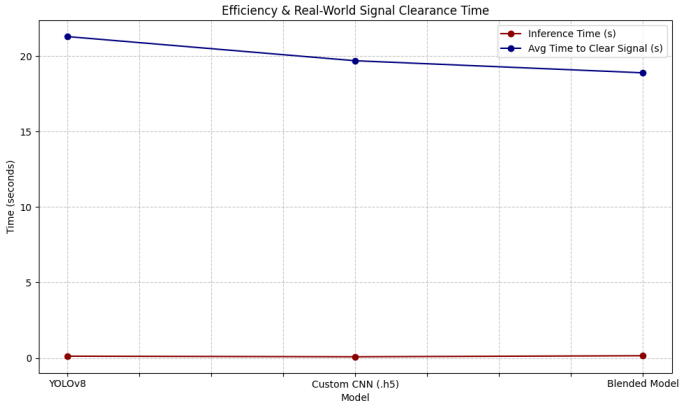

Fig. 14: Efficiency and Real-world Signal Clearance Time per Model

IX. Result Visualization

F. Visual Evaluation

This section provides both visual and performance-based evaluation. Figure 13 shows the real-time vehicle detection and classification output. Additionally, Figure 14 compares the average time taken to clear a signal using the proposed method.

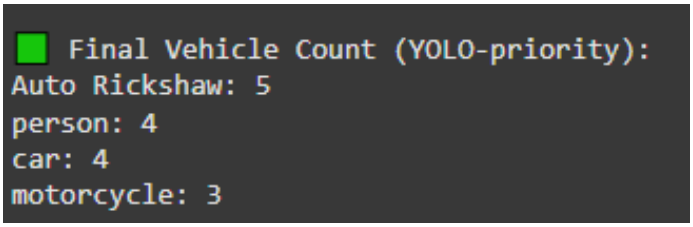

Fig. 15: Detected vehicle count with YOLO as priority and then pass on undetected object to Indian vehicle classification

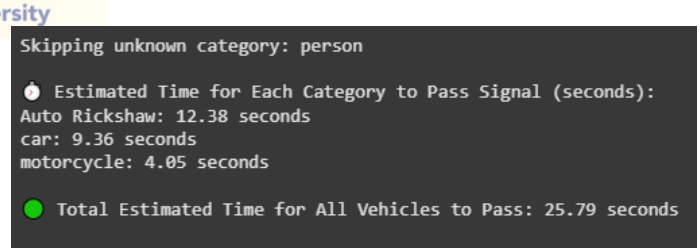

Fig. 16: Detected vehicle count with estimated time to clear the signal (e.g., 35 seconds for 12 vehicles)

X. Impact & Application

- The proposed hybrid approach effectively detects and classifies both general and region-specific Indian vehicles, ensuring high accuracy in complex traffic scenarios.
- It assists **smart traffic signal systems** to dynamically allocate green signal time based on real-time vehicle count and type (e.g., auto-rickshaw vs. truck).
- The system supports **urban traffic planning**, congestion reduction, and emergency routing decisions in Indian cities.
- Can be deployed in **smart cities**, **automated surveillance**, and **toll automation** systems for intelligent transportation management.

XI. Room for Improvement

- **Dataset Expansion**: Incorporating more diverse images (e.g., nighttime, rain, occlusions) to improve model robustness.
- **Model Integration**: Training a unified YOLO model with Indian vehicle classes could reduce inference time.
- **Video Stream Optimization**: Enhancing real-time processing capabilities for live traffic footage, especially on edge devices.
- **Accuracy Boost**: Integrating ensemble learning or transformer-based models to improve classification performance.
- **Multi-Lane Handling**: Extending the model to analyze and handle multiple lanes independently.

XII. Conclusion

This project presents an efficient two-stage deep learning pipeline combining **YOLOv8** for object detection and a **custom-trained CNN** for Indian vehicle classification. By leveraging real-time detection and intelligent post-classification, the system estimates the time required to clear the traffic signal based on vehicle type and count. The model demonstrates strong real-world applicability in smart traffic systems and urban mobility planning, with a promising path for future enhancement.

References

References

[1] Jocher, G. et al. (2023). YOLO by Ultralytics. GitHub Repository. https://github.com/ultralytics/ultralytics.
[2] Mihir M Gandhi. Adaptive Traffic Signal Timer. https://github.com/mihir-m-gandhi/Adaptive-Traffic-Signal-Timer.
[3] Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4), 600–612.
[4] Sharma, S., Jain, A., & Singh, A. (2021). Intelligent Traffic Light Control Using Real-Time Vehicle Detection and Classification. *International Journal of Computer Applications*, 183(5), 1–6.
[5] Brownlee, J. (2019). *Deep Learning for Computer Vision*. Machine Learning Mastery.
[6] Chollet, F. (2015). Keras: Deep Learning for Humans. https://keras.io.
[7] OpenCV. (n.d.). Open Source Computer Vision Library. https://opencv.org.
[8] TensorFlow. (n.d.). An end-to-end open source machine learning platform. https://www.tensorflow.org.