Intelligent Traffic Management System on NVIDIA Jetson Nano

submitted in partial fulfillment of the requirements

for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINNEERING

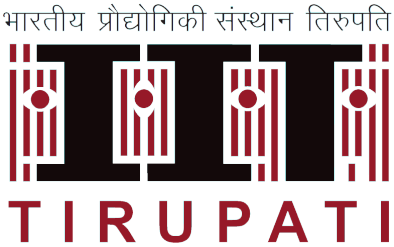
by

CHATSE SIDDHANT MADHUKAR CS22B016

HARSHIT GARG CS22B024

Supervisor(s)

Dr. Jayanarayan T Tudu



DEPARTMENT OF COMPUTER SCIENCE AND ENGINNEERING

INDIAN INSTITUTE OF TECHNOLOGY TIRUPATI

OCTOBER 2025

DECLARATION

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission to the best of my knowledge. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Place: Tirupati
Date: 29-11-2025

Signature
CHATSE SIDDHANT MADHUKAR
CS22B016

Place: Tirupati
Date: 29-11-2025

Signature
HARSHIT GARG
CS22B024

BONA FIDE CERTIFICATE

This is to certify that the report titled **Intelligent Traffic Management System on NVIDIA Jetson Nano**, submitted by **CHATSE SIDDHANT MADHUKAR** and **HARSHIT GARG**, to the Indian Institute of Technology, Tirupati, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the project work done by him [or her] under our or [my] supervision. The contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Place: Tirupati
Date: 29-11-2025

Dr. Jayanarayan T Tudu
Guide
Assistant Professor
Department of Computer Science and Engineering
IIT Tirupati - 517501

ABSTRACT

Traditional fixed timer traffic signals are inefficient in handling dynamic and heterogeneous urban traffic. This project presents an intelligent, vision-based adaptive traffic signal control system that allocates green signals based on real-time traffic conditions. A YOLOv5-based object detection model was fine-tuned on the Indian Driving Dataset to detect multiple vehicle classes under complex traffic scenarios and optimized using TensorRT for real-time inference on an NVIDIA Jetson Nano. Lane-wise congestion is estimated using weighted vehicle counts, and a fair signal decision strategy incorporating waiting time is employed to prevent lane starvation. Signal transitions are synchronized at fixed intervals, and a real-time visualization provides continuous system monitoring. The results demonstrate reliable, fair, and temporally stable traffic signal decisions, validating the system's effectiveness for real-world deployment in Indian traffic environments.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

BDD	Batch-Based Detection
BTP	Bachelor Thesis Project
CUDA	Compute Unified Device Architecture
CNN	Convolutional Neural Network
CV	Computer Vision
FPS	Frames Per Second
GPU	Graphics Processing Unit
IDD	Indian Driving Dataset
IoU	Intersection over Union
NMS	Non-Maximum Suppression
ONNX	Open Neural Network Exchange
PT	PyTorch Model File
RT	Real-Time
SNR	Signal-to-Noise Ratio
TRT	TensorRT
YOLO	You Only Look Once
XML	Extensible Markup Language

NOTATION

nc	Number of classes in the dataset.
$names$	List of class names corresponding to class IDs.
$path$	Root directory path for the dataset.
$train$	Path to the directory or text file for training images.
val	Path to the directory or text file for validation images.
$labels$	Root directory containing label subfolders.
P	Precision metric.
R	Recall metric.
$mAP@.50$	Mean Average Precision at an IoU threshold of 50%.
$mAP@.50–.95$	Mean Average Precision averaged over IoU thresholds from 50% to 95%.
I	Input image or video frame.
B	Predicted bounding box coordinates (x_1, y_1, x_2, y_2).
C	Predicted object class label.
$Conf$	Confidence score associated with a detected object.
IoU	Intersection over Union between predicted and ground-truth boxes.
L_i	Index of traffic lane i, where $i \in \{1, 2, 3, 4\}$.
N_i	Number of detected vehicles in lane i.
w_j	Weight assigned to vehicle class j based on its traffic impact.
C_i	Lane-wise congestion value for lane i.
t	Current system time.
$t_{\textbf{green},i}$	Last time at which lane i received a green signal.
W_i	Waiting time of lane i since its last green signal.
T_s	Fixed signal change interval (15 seconds).
E_i	Effective congestion score for lane i.
α	Weight factor assigned to congestion contribution (0.7).
β	Weight factor assigned to waiting time contribution (0.3).
Δt_d	Object detection interval (3 seconds).
Δt_s	Signal scheduling interval (15 seconds).

CHAPTER 1

INTRODUCTION

1.1 Objectives and Scope

Objective

The primary objective of this project is to design and implement an intelligent, vision-based adaptive traffic signal control system capable of dynamically allocating green signal time based on real-time traffic conditions, with a strong emphasis on fairness, robustness, and deployability on edge hardware.

- To fine-tune a deep learning–based object detection model (YOLOv5) on traffic datasets representative of Indian road conditions, enabling accurate multi-class vehicle detection in real-world scenarios.
- To develop a computer vision pipeline capable of performing real-time vehicle detection across multiple lanes using CCTV-like video feeds.
- To estimate lane-wise traffic congestion using weighted vehicle counts, accounting for the varying impact of different vehicle categories (e.g., cars, buses, trucks, two-wheelers).
- To design and implement a fair and adaptive traffic signal scheduling strategy that integrates both congestion levels and waiting time, thereby preventing lane starvation and ensuring equitable signal distribution.
- To deploy and optimize the complete system on edge hardware (NVIDIA Jetson Nano) under real-time constraints, utilizing inference acceleration techniques.
- To analyze the performance, stability, and correctness of signal decisions under varying traffic densities and dynamic conditions, validating the effectiveness of the proposed control logic.

Scope

This project focuses on designing and implementing a real-time, vision-based adaptive traffic signal control system for a four-lane road intersection. The system processes live or recorded video streams from fixed cameras to detect and classify vehicles using a deep learning–based object detection model. YOLOv5 is fine-tuned for Indian traffic conditions and optimized using TensorRT to enable efficient inference on edge hardware. Lane-wise traffic congestion is estimated through weighted vehicle counts, where different vehicle types contribute differently to overall congestion levels.

Based on congestion estimates and the waiting time since a lane last received a green signal, an adaptive signal control algorithm dynamically allocates the green phase to the most eligible lane while preventing starvation of low-traffic lanes. The complete pipeline including video preprocessing, inference, congestion computation, and signal decision logic is deployed and evaluated on an NVIDIA Jetson Nano platform under real-time constraints. The scope of this work excludes city-level traffic coordination, direct hardware signal interfacing, Ambulance and Fire Brigade Vehicle prioritization, which are considered future extensions.

CHAPTER 2

Literature Review

2.1 Introduction

Traffic congestion has become a major challenge in urban areas due to the increasing number of vehicles and inefficient traffic control systems. Traditional fixed-timer systems fail to adapt to real-time variations in traffic flow, leading to unnecessary delays and fuel wastage. Recent advancements in computer vision and deep learning have paved the way for Intelligent Traffic Management Systems (ITMS) that use live video feeds and detection algorithms such as YOLO and Faster R-CNN for real-time vehicle detection, traffic density estimation, and adaptive signal control. This chapter reviews four significant research papers that have contributed to the development of vision-based and machine learning–driven traffic management systems.

2.2 Vision-Based Intelligent Traffic Light Management System Using Faster R-CNN[1]

Abbas and others (2024) proposed a vision-based adaptive traffic light system employing the Faster R-CNN algorithm for vehicle detection and classification. The model was trained on CCTV datasets and achieved 95.7% detection and 96.6% classification accuracy, outperforming YOLOv2 and other models. By dynamically adjusting signal durations based on vehicle density, the system improved road throughput and minimized idle time. The study highlights the effectiveness of region proposal networks in achieving high precision, making it suitable for smart city deployment.

2.3 Intelligent Traffic Management System Using Computer Vision and Machine Learning[4]

Sakhuja (2023) developed a complete AI-based traffic management framework integrating computer vision and machine learning for congestion control and incident detection. The proposed system uses real-time camera feeds to detect vehicles, analyze patterns, and optimize signal timing through predictive analytics. Additionally, it includes a GSM-based alert system for emergency situations. This paper demonstrates the importance of combining vision-based perception with data-driven decision-making for efficient traffic regulation.

2.4 Intelligent Traffic Monitoring and Management Based on Computer Vision Technology[2]

Chunmei Chen (2024) presented a system that integrates YOLO object detection with Kalman filtering to track vehicles and predict motion. The framework processes road footage and environmental data to provide accurate flow estimation and anomaly detection. The system reduced average diversion time by 48 minutes and accident rate by 0.3%, proving the reliability of computer vision–based monitoring. This paper shows the potential of integrating motion prediction and environmental awareness for smarter, safer intersections.

2.5 Computer Vision–Based Intelligent Traffic Management System[3]

Darwhekar and others (2022) introduced a YOLOv3-based real-time detection system for adaptive traffic control. The approach uses Non-Maximum Suppression to refine detections and calculates vehicle counts to adjust green-light durations dynamically. The model achieved 145 fps processing speed, ensuring near-instant responsiveness. Its practical, low-cost approach is ideal for developing urban infrastructures that require fast deployment and minimal hardware resources.

CHAPTER 3

Intelligent Traffic Management System

3.1 About this Chapter

This chapter presents the detailed design and implementation of the proposed Intelligent Traffic Management System. It explains the core components of the system architecture, including video acquisition, vehicle detection, lane-wise congestion estimation, and adaptive signal control logic. Emphasis is placed on how real-time traffic information is extracted from visual data and how this information is transformed into effective signal control decisions under practical constraints.

The chapter further describes the mathematical foundations, algorithms, and optimization techniques used to ensure fairness, efficiency, and real-time feasibility. It also discusses the deployment considerations on edge hardware, justified design choices, and system-level integration of computer vision and decision logic. Through this chapter, the complete operational pipeline of the intelligent traffic control system is established, forming the basis for experimental evaluation and performance analysis in subsequent chapters.

3.2 Mathematical Formulation

3.2.1 Bounding Box Regression

The model predicts the coordinates of a bounding box (b_x, b_y, b_w, b_h), representing the center (x, y), width, and height of a detected object. The loss for this prediction is typically calculated using a Complete IoU (CIoU) loss, which is an advanced form of Intersection over Union (IoU).

$$\text{IoU} = \frac{|B_p \cap B_{gt}|}{|B_p \cup B_{gt}|}$$

The CIoU loss function adds penalties for the center point distance and aspect ratio differences, leading to faster and more stable convergence than simple L2 loss.

3.2.2 Confidence and Class Prediction

For each predicted box, the model outputs a confidence score (P_c) and a vector of class probabilities ($C_1, C_2, ..., C_n$).

$$\text{Final Score}_{\text{class } i} = P_c \times C_i$$

The model's loss function combines the bounding box regression loss, objectness confidence loss, and class probability loss into a single value minimized during training.

3.2.3 Vehicle Detection and Congestion Estimation

Let L = {1,2,3,4} denote the set of traffic lanes. Each lane is continuously monitored using video streams, and vehicles are detected using a deep learning based object detection model.

For each lane $i \in L$, the object detection model identifies vehicles along with their respective class labels. Since different vehicle types contribute unequally to traffic congestion, a weighted congestion estimation approach is adopted. Let $N_{i,c}$ denote the number of detected vehicles of class c in lane i, and let w_c represent the weight assigned to class c. The congestion score for lane i is defined as:

$$C_i = \sum_c w_c \cdot N_{i,c}$$

This formulation ensures that larger vehicles such as buses and trucks contribute more to the congestion score than smaller vehicles like motorcycles.

3.2.4 Waiting Time Modeling

To prevent starvation of any lane, the system incorporates a waiting-time based fairness mechanism. Let t denote the current signal decision time, and let t_i^{last} be the last time at which lane i was assigned a green signal. The waiting time for lane i is computed as:

$$W_i(t) = t - t_i^{\text{last}}$$

If a lane has never received a green signal, its last green time is initialized to the system start time, ensuring that all lanes have a waiting time of zero at $t = 0$.

3.2.5 Effective Priority Score

To balance congestion mitigation and fairness, an effective priority score is computed for each lane by combining congestion and waiting time. The effective score E_i for lane i is given by:

$$E_i = \alpha \cdot C_i + \beta \cdot W_i$$

where α and β are weighting coefficients such that: $\alpha = 0.7, \quad \beta = 0.3, \quad \alpha + \beta = 1.$

This ensures that congestion remains the dominant factor while progressively increasing the priority of lanes that have waited longer.

3.2.6 Signal Selection Criterion

At fixed decision intervals, the lane with the highest effective priority score is selected for receiving the green signal:

$$i^* = \arg\max_{i \in L} E_i$$

Once lane i^* is selected, its last green time is updated according to:

$$t_{i^*}^{\text{last}} = t$$

This update resets the waiting time for the selected lane while allowing the waiting times of other lanes to increase.

3.2.7 Timing Constraints

To ensure signal stability and avoid rapid oscillations between lanes, the system enforces a minimum green-time constraint. A signal change is permitted only if:

$$t - t^{\text{change}} \geq T_{\min}$$

where t^{change} is the time of the previous signal decision, and $T_{\min}$ is the minimum allowable green duration.

This mathematical formulation enables a stable, fair, and congestion-aware traffic signal control mechanism suitable for real-time deployment.

3.3 Experimental Setup

- **Training System (Workstation):** HP Z2 Tower G9, Intel i7-12700, RTX 3050 (8GB), 32GB RAM, Ubuntu 24.04.
- **Deployment Hardware:** NVIDIA Jetson Nano (4GB RAM).
- **Development Environment:** Visual Studio Code, Python 3.8 (Jetson) and Python 3.12 (workstation), PyTorch, Ultralytics YOLOv5 (v7.0), Nvidia JetPack SDK 4.6.1, DeepStream 6.0.1, OpenCV, NumPy, PyCUDA, TensorRT.
- **Object Detection Model:** YOLOv5s architecture fine-tuned for traffic vehicle detection under Indian road conditions.
- **Dataset:** Indian Driving Dataset (IDD) – Detection split (22.8GB, 40,000 images).
- **Model Optimization Pipeline:** PyTorch $\rightarrow$ ONNX $\rightarrow$ TensorRT conversion with FP16 optimization for low-latency inference on Jetson Nano.
- **Video Inputs:** Four pre-recorded traffic videos, each representing an independent lane at a junction, with varying vehicle densities.

3.4 Procedure, Techniques and Methodologies

The development of the proposed Intelligent Traffic Management System followed a structured, multi-stage methodology to ensure correctness, reproducibility, and deployment readiness. The overall workflow of the system, illustrated in Figure 3.1, was designed to systematically transform raw traffic data into a real-time deployable solution.

The workflow begins with the acquisition of the Indian Driving Dataset (IDD), consisting of road scene images and corresponding annotations. A dedicated preprocessing phase was carried out to reorganize the dataset and convert the annotations into YOLO-compatible text format using a custom conversion pipeline. Following data preparation, the detection model was configured using a YAML-based specification and trained on a GPU-enabled workstation. Model performance was iteratively evaluated, and the best-performing checkpoint (`best.pt`) was selected. This trained model was subsequently exported and optimized for edge deployment, forming the foundation for subsequent real-time inference, congestion estimation, and adaptive signal control stages described in the following phases.

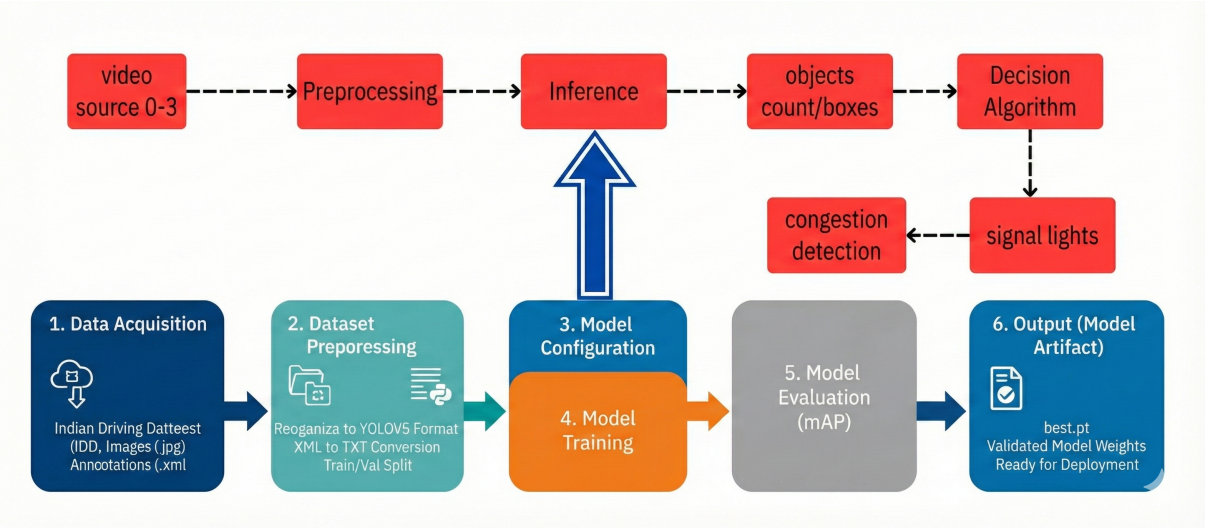


Figure 3.1: Project Workflow

Phase 1: Environment Setup

Cloned the YOLOv5 repository and set up dependencies in a virtual environment. This step involved cloning the official Ultralytics YOLOv5 repository from GitHub to the workstation. A dedicated Python virtual environment was created using `venv` to manage all required libraries, such as PyTorch, NumPy, and OpenCV, preventing conflicts with system packages.

Phase 2: Dataset Preparation and Preprocessing

Converted Pascal VOC (.xml) annotations to YOLO format using a custom Python script. The raw IDD dataset provided annotations in an `.xml` (Pascal VOC) format, which is incompatible with the YOLOv5 training script. A custom Python script (`XML_to_YOLO_Converter_Split.py`)

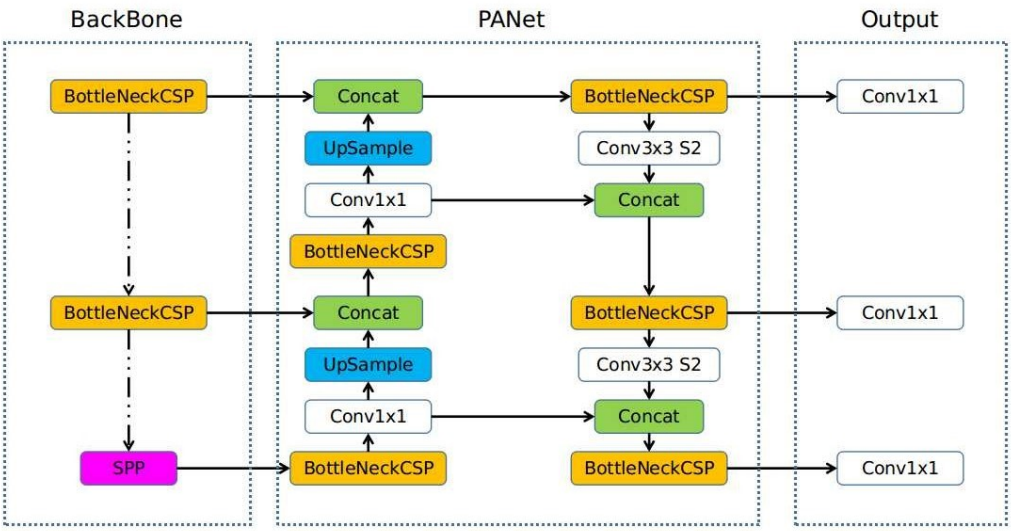


Figure 3.2: YOLO v5 architecture

was developed to parse these XML files, extract bounding box coordinates and class names, and convert them into the required normalized YOLO `.txt` format. This process was run for all images listed in the `train.txt`, `val.txt`, and `test.txt` split files, mirroring the dataset's complex subdirectory structure.

Phase 3: Model Configuration

Created `idd_yolo_dataset.yaml` defining paths, class names, and count. A YAML configuration file is required by YOLOv5 to understand the dataset's structure. This file, `idd_yolo_dataset.yaml`, was created to define the absolute paths to the `images` and `labels` directories, specify the exact image files for training and validation using the `train.txt` and `val.txt` files, and declare the number of classes (`nc: 6`) along with their ordered names.

Phase 4: Model Training

Used pretrained weights (`yolov5s.pt`) for transfer learning. To accelerate training and achieve higher accuracy, we employed transfer learning. The `train.py` script was initiated using the `yolov5s.pt` pre-trained weights, which were trained on the large COCO dataset. This allowed the model to leverage its existing knowledge of general object features and fine-tune its final layers for our specific IDD dataset, resulting in faster convergence and a more robust final model.

Phase 5: Model Optimization and Deployment Preparation

The fine-tuned YOLOv5 model, initially trained using the PyTorch framework, was exported to the ONNX (Open Neural Network Exchange) format. This intermediate representation ensures framework independence and serves as a bridge for hardware-specific optimization. Subsequently, the ONNX model was converted into a TensorRT engine with FP16 precision to significantly reduce inference latency and memory usage, making it suitable for execution on the NVIDIA Jetson Nano edge device.

Phase 6: Multi-Lane Inference and Vehicle Detection

Four independent video streams, each corresponding to a traffic lane, were processed simultaneously. Frames were resized to 640×640, normalized, and grouped into a single batch for inference. This batch-based processing enabled a single forward pass through the TensorRT engine to handle all lanes efficiently, ensuring real-time performance on edge hardware.

The inference outputs were decoded to obtain vehicle bounding boxes, class labels, and confidence scores. Lane-wise Non-Maximum Suppression (NMS) reduced duplicate detections, and the final vehicle counts were converted into congestion values using predefined class-wise weights(Table 3.1), assigning higher importance to heavy vehicles such as buses and trucks.

Class Name	**Class ID**	**Weight** (w_i)
Car	0	1.0
Autorickshaw	1	1.0
Truck	2	3.0
Motorcycle	3	0.5
Bus	4	3.0

Table 3.1: Vehicle Class Weights Used for Lane Congestion Estimation

Phase 7: Lane-Wise Congestion Estimation and Fair Signal Decision Logic

Lane-wise congestion was estimated using a *weighted vehicle count* instead of a simple vehicle count to accurately model heterogeneous Indian traffic conditions. Each detected vehicle contributed to lane congestion according to its relative impact on road occupancy and traffic flow. Heavy vehicles such as buses and trucks were assigned higher weights, while lighter vehicles such as motorcycles were assigned lower weights.

The congestion score for a lane L at time t was computed as:

$$C_L(t) = \sum_{i=1}^{N_L(t)} w_{c_i}$$

where $N_L(t)$ denotes the number of vehicles detected in lane L at time t, and w_{c_i} represents the congestion weight assigned to the i^{th} detected vehicle belonging to class c_i (listed in Table 3.1).

To ensure fairness and avoid lane starvation, a wait-time–aware signal control strategy was employed. For each signal decision interval, an *effective congestion score* was calculated for every lane by combining both current congestion and waiting time:

$$E_L(t) = \alpha C_L(t) + (1-\alpha) W_L(t)$$

where $W_L(t)$ is the time elapsed since lane L last received a green signal, and α is a weighting coefficient. In this work, $\alpha = 0.7$ was chosen to prioritize real-time congestion while still ensuring fairness. During every signal update cycle, the lane with the maximum effective score $E_L(t)$ was selected to receive the green signal, thereby balancing throughput efficiency and starvation prevention.

Phase 8: Time-Synchronized Signal Scheduling and Real-Time Visualization

To ensure temporal consistency and eliminate decision drift caused by processing delays, signal transitions were executed at exact multiples of a fixed signal interval of 15 seconds. Signal evaluation and switching strictly followed this synchronized schedule (e.g., 0s, 15s, 30s, ...), independent of frame rate or inference latency. Additionally, a minimum green time constraint of 15 seconds was enforced to prevent rapid signal oscillations under fluctuating traffic conditions, thereby improving system stability and intersection safety.

A real-time visualization and monitoring module was developed using OpenCV to display the system's operational state. Video feeds from all four lanes were rendered in a 2×2 tiled layout, where each tile displayed detected vehicles with bounding boxes, lane number, current signal status (GREEN/RED), and the effective congestion score. A centrally positioned countdown timer indicated the remaining time until the next signal change. This visualization framework proved essential for debugging, validating decision correctness, and conducting qualitative performance analysis during experimental evaluation.

CHAPTER 4

Results and Discussions

The model training was conducted for 100 epochs using the fine-tuned YOLOv5s model on the IDD dataset. The training process successfully completed, and the final validation metrics were automatically computed by the training script.

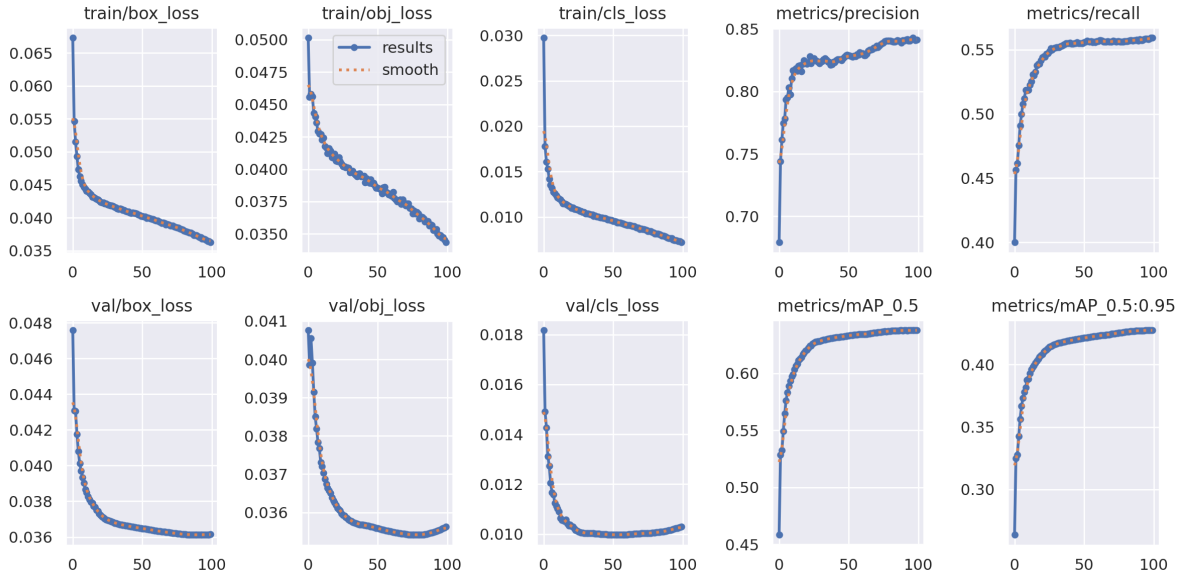


Figure 4.1: Results

Figure 4.1 shows the learning curves for the model, indicating a steady decrease in loss and an increase in mAP over time.

The validation results provide a quantitative measure of the model's performance on unseen data. The key metrics evaluated include Precision (P), Recall (R), and Mean Average Precision (mAP) at different Intersection over Union (IoU) thresholds.

- **Precision (P):** "Of all the detections our model made, what percentage was correct?" A high precision means the model produces few false positives, so it doesn't "invent" vehicles that aren't there.
- **Recall (R):** "Of all the actual vehicles present in the images, what percentage did our model find?" A lower recall means the model is missing a significant number of true objects.

- **mAP@.50 (mAP50):** The primary metric for object detection. It measures the average precision across all classes, considering a detection "correct" if its bounding box overlaps the true box by at least 50% (IoU > 0.5). This is a good measure of the model's general detection quality.
- **mAP@.50-.95 (mAP50-95):** A much stricter metric that averages the mAP across 10 different IoU thresholds (from 50% to 95%). A lower score here (0.428) compared to mAP50 (0.638) is normal and indicates the model is good at finding objects (mAP50) but less precise at perfectly aligning the bounding boxes (mAP50-95).

Class	Images	Instances	P	R	mAP@.50	mAP@.50-.95
all	10225	88185	0.841	0.559	0.638	0.428
car	10225	24844	0.870	0.575	0.662	0.460
autorickshaw	10225	7782	0.871	0.604	0.689	0.487
truck	10225	7077	0.798	0.552	0.629	0.442
motorcycle	10225	25489	0.851	0.589	0.659	0.391
bus	10225	4915	0.850	0.634	0.703	0.525
person	10225	18078	0.805	0.402	0.484	0.260

Table 4.1: Model Validation Performance (yolov5s)

While the above results focused on model training and validation metrics on the IDD dataset, the work also includes analysis to real-time deployment aspects, including TensorRT-accelerated inference, lane-wise congestion estimation, fairness-aware signal decision logic, and time-synchronized signal scheduling.

The behavior of the system was evaluated using multi-lane traffic video streams, with emphasis on detection reliability, correctness of signal decisions, temporal stability, and fairness across lanes. Console outputs and real-time visualizations were jointly used to assess whether the system's decisions aligned with observed traffic conditions.

4.1 Analysis of Results

1. **Overall Performance:** The overall mAP@.50 score of **0.638** is a respectable result, considering the complexity of the 6-class dataset and the challenging, often chaotic nature of the IDD images. The high overall Precision (0.841) is excellent, as it means the data fed into the congestion algorithm will be reliable, with few "ghost" detections. The main

weakness is the lower Recall (0.559), which indicates the model misses approximately 44% of the objects in the validation set.

2. **Per-Class Performance:**
    - **Strongest Classes:** ‘bus‘ (0.703 mAP@.50) and ‘autorickshaw‘ (0.689 mAP@.50) performed the best. This is likely due to their large and/or highly distinct visual features, making them easier for the model to identify. The high recall for ‘bus‘ (0.634) is particularly good.
    - **Average Classes:** ‘car‘ (0.662 mAP@.50) and ‘motorcycle‘ (0.659 mAP@.50) performed well and in line with the average. Given that motorcycles are often small and heavily occluded, this result is strong.
    - **Weakest Class:** ‘person‘ (0.484 mAP@.50) was the weakest. The ‘person‘ class has very low Recall (0.402) and a low mAP50-95 (0.260). This is expected, as people are small, often occluded by vehicles, and have highly variable appearances, making them the most difficult class to detect in complex traffic scenes. This is further visualized in the confusion matrix (Figure 4.2).

The Precision-Recall curve (Figure 4.3) further illustrates the model’s performance, showing the trade-off for each class.

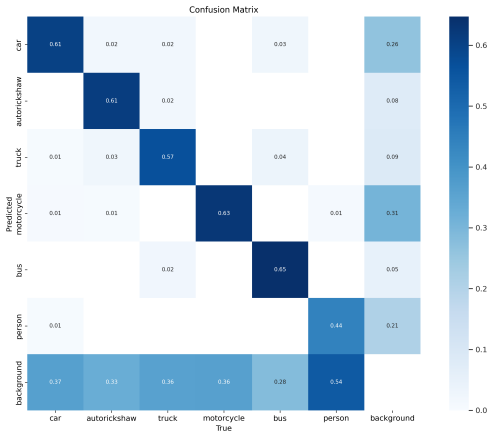


Figure 4.2: Confusion Matrix

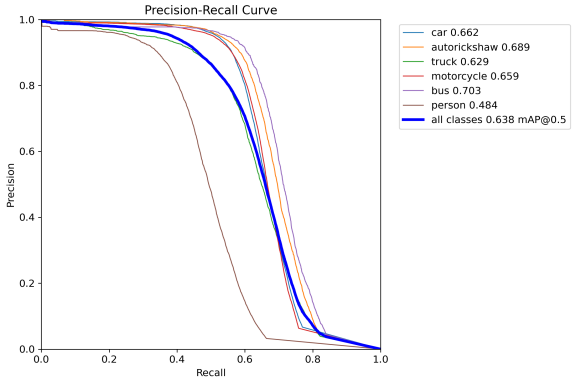


Figure 4.3: PR Curve

A visual representation of the model’s inference capabilities is provided in Figure 4.4, showcasing real-time detections on a sample traffic video frame from the IDD.

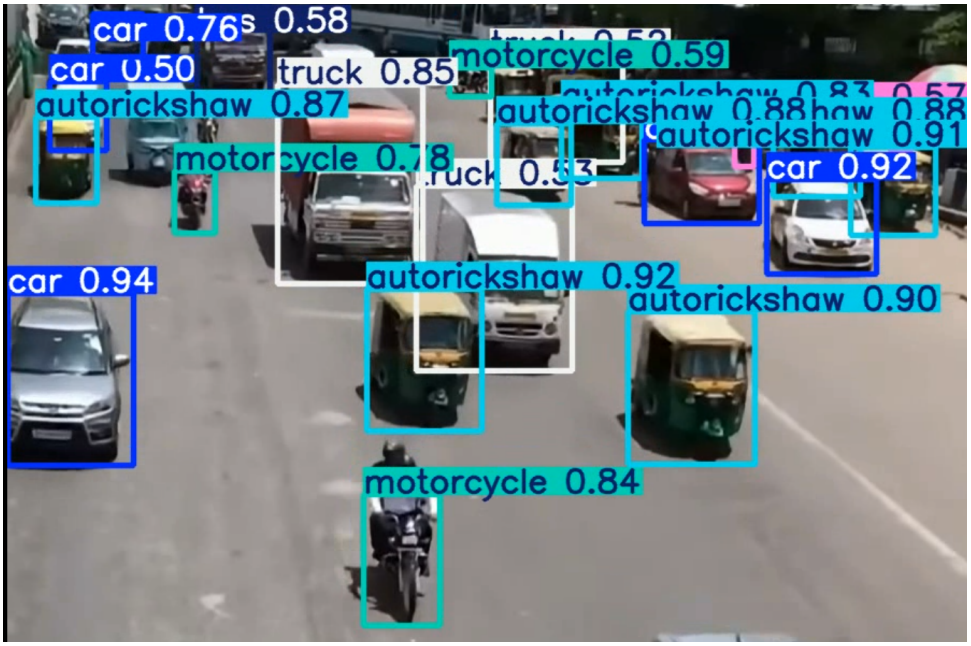


Figure 4.4: Sample Real-time Object Detection on an Indian Traffic Frame

3. **Signal Decision Behavior and Correctness:** The system initiates with an immediate detection phase followed by the first signal decision at t=0 seconds. At this instant, all lanes have zero waiting time; therefore, signal allocation is governed solely by weighted congestion values. As observed in the execution results (Figure 4.5), the lane with the highest congestion consistently achieved the maximum effective score and was correctly assigned the green signal.

```
(btp_env) nvida@nvida-desktop:~/btp_project$ python3 video_trt_yolo_4lanes.py --engine /home/nvida/
btp_project/yolov5s_idd_bs4.engine --videos vid1.mp4 vid2.mp4 vid3.mp4 vid4.mp4
Detection every 3.0s | Signal change interval: 15.0s
Engine input shape: (4, 3, 640, 640)
Initial detection + immediate first decision at 0s.
[detect] ran at 4.59s  cong=[30.5, 9.5, 17.0, 5.0]

===== SIGNAL (decision time) @ 0s =====
Lane 1: GREEN | cong=30.50 | wait= 0.00 | effective=21.35
Lane 2: RED   | cong= 9.50 | wait= 0.00 | effective= 6.65
Lane 3: RED   | cong=17.00 | wait= 0.00 | effective=11.90
Lane 4: RED   | cong= 5.00 | wait= 0.00 | effective= 3.50
========================================
*** DECISION: lane 1 GIVEN GREEN (based on pre-update effective). ***
(post) Lane 1: wait= 0.00 | effective=21.35
(post) Lane 2: wait= 0.00 | effective= 6.65
(post) Lane 3: wait= 0.00 | effective=11.90
(post) Lane 4: wait= 0.00 | effective= 3.50
========================================
[detect] ran at 6.06s  cong=[33.5, 13.0, 13.0, 9.0]
[detect] ran at 9.03s  cong=[35.0, 6.5, 14.0, 9.0]
[detect] ran at 12.02s  cong=[24.0, 9.0, 11.0, 7.0]
[detect] ran at 15.05s  cong=[31.5, 12.0, 23.0, 4.0]
```

Figure 4.5: Real-time signal decision based on effective congestion scores across four lanes.

At subsequent decision intervals (every 15 seconds), the system re-evaluates lane priorities using updated congestion and accumulated waiting times. The selected green lane always corresponds to the maximum effective congestion score calculated before updating signal

states, confirming correctness of the decision logic and eliminating false priority inversion.

4. **Temporal Consistency and Signal Scheduling:** To ensure deterministic behavior, signal decisions are executed strictly at exact multiples of the predefined signal interval of 15 seconds. This approach eliminates timing drift caused by processing and inference delays.

   While vehicle detection runs every 3 seconds to maintain up-to-date congestion estimates, signal changes are restricted to the fixed scheduling grid and enforced with a minimum green duration of 15 seconds. This guarantees temporal stability, prevents rapid oscillations, and ensures that each green phase allows meaningful traffic clearance.

5. **Fairness and Lane Starvation Prevention:** Unlike fixed-time or congestion-only strategies, the final system integrates waiting time into the signal decision logic. As lanes remain in the red state, their waiting time increases linearly, causing their effective congestion score to rise even if vehicle density remains moderate.

   This behavior is visible in later execution logs, where lanes with prolonged red durations gradually become competitive and are eventually assigned green. This confirms that the system avoids lane starvation and ensures fairness across all approaches, a critical requirement for real-world Indian intersections.

6. **Visual Validation of System Output:** The real-time visualization module provides strong qualitative validation of system performance. As shown in Figure 4.6, each lane is displayed in a 2×2 tiled layout with:
   - Detected vehicles enclosed by bounding boxes,
   - Lane number and current signal status (GREEN/RED),
   - Effective congestion score displayed per lane,
   - A centrally placed countdown timer indicating time remaining until the next signal change.

   The effective congestion values rendered on-screen match the values computed and logged internally, confirming synchronization between backend decision logic and frontend display.

   Additionally, the currently green lane is highlighted in the same color as the countdown timer, improving interpretability and providing a clear indication of signal priority during execution.

Figure 4.6: Real-time multi-lane traffic visualization showing vehicle detection, lane-wise signal status, effective congestion scores, and synchronized signal countdown.

4.2 Implications for Deployment

The experimental results demonstrate that the proposed system is well-suited for real-time deployment in adaptive traffic signal control scenarios. The high precision achieved by the fine-tuned YOLOv5s model (0.841 overall) ensures that vehicle detections are reliable, minimizing false positives. This is critical in deployment, as inaccurate detections could otherwise lead to incorrect congestion estimation and inefficient signal allocation for empty or lightly loaded lanes.

Although the recall is relatively moderate (0.559), particularly for smaller or occluded objects such as motorcycles and pedestrians, the system has been explicitly designed to remain robust under such underestimation. The deployment pipeline incorporates weighted congestion modeling and a wait-time–aware signal decision mechanism, ensuring that signal choices are not solely dependent on instantaneous vehicle counts. Additionally, the inclusion of starvation-prevention logic by factoring in lane waiting time guarantees that lanes with missed detections are still periodically served. With TensorRT-accelerated inference enabling real-time performance on edge hardware, the system is validated as deployment-ready, with its known limitations

effectively mitigated through algorithmic design.

CHAPTER 5

SUMMARY AND CONCLUSION

5.1 Summary and Conclusion

This project successfully evolved from object detection model development to a fully integrated, real-time Intelligent Traffic Management System. In the initial phase, a YOLOv5s model fine-tuned on the Indian Driving Dataset (IDD) achieved strong detection performance with an mAP@0.50 of 0.638 and high precision, making it reliable for traffic analysis in Indian road conditions.

Building on this model, a real-time adaptive traffic control pipeline was developed. The model was optimized using ONNX and TensorRT for low-latency edge deployment, and four lane-wise video streams were processed concurrently using batch inference. Traffic congestion was estimated using weighted vehicle counts, and a wait-time–aware signal control strategy ensured fair and starvation-free signal allocation at fixed 15-second intervals.

The system demonstrated stable, interpretable, and correct signal decisions under varying traffic densities through real-time visualization and synchronized signal scheduling. Overall, the work establishes a practical and deployable vision-based traffic signal control framework, providing a strong foundation for future real-world deployment and further optimization.

5.2 Scope for Future Work

The current system provides a strong software-level validation of vision-based adaptive traffic control. The scope of future work is as follows:

1. **Hardware Integration:** Integrate the system with physical traffic signal controllers using the GPIO or industrial interfaces of Jetson Nano to enable real-time actuation of signal lights at actual junctions.

2. **Emergency Vehicle Priority Handling:** Extend the system to detect and prioritize emergency vehicles such as ambulances and fire brigades, enabling immediate green signal assignment to the corresponding lane.

REFERENCES

[1] **S. K. Abbas**, **M. U. Hassan**, **R. Sarwar**, *et al.* (2024). Vision based intelligent traffic light management system using faster–r-cnn. *CAAI Transactions on Intelligence Technology*. URL https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/cit2.12309.

[2] **C. Chen** (2024). Intelligent traffic monitoring and management based on computer vision technology. *IEEE*. URL https://ieeexplore.ieee.org/document/10594478.

[3] **S. G. R. B. S. R. Kshitij Darwhekar, Amey Patil** (2022). Computer vision based intelligent traffic management system. *IEEE*. URL https://ieeexplore.ieee.org/document/10009105.

[4] **A. Sakhuja** (2023). Intelligent traffic management system using computer vision and machine learning. *Innovative Research Thoughts/ResearchGate*. URL https://www.researchgate.net/publication/374717408_Intelligent_Traffic_Management_System_using_Computer_Vision_and_Machine_Learning.