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

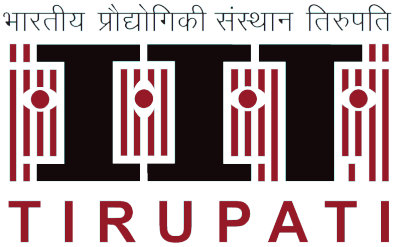
by

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: 31-10-2025

Signature
CHATSE SIDDHANT MADHUKAR
CS22B016

Place: Tirupati
Date: 31-10-2025

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: 31-10-2025

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

Note: If supervised by more than one professor, professor's name must be included and get signatures from all supervisors. Change him/her or our or my accordingly.

ABSTRACT

Urban traffic congestion is a critical problem driven by increasing vehicle density. Traditional, fixed-timer traffic signals are inefficient and fail to adapt to real-time traffic dynamics. This project details the design and training of a computer vision model, the first phase in developing an Intelligent Traffic Management System. We create a custom object detection model by training YOLOv5 on the specialized Indian Driving Dataset (IDD). This report covers the complete machine learning workflow: dataset preparation, annotation conversion, model training on a GPU-enabled workstation, and a detailed analysis of the resulting model's performance. The final model achieves a respectable mAP@.50 of 0.638 on the complex 6-class dataset, demonstrating its capability to accurately identify diverse vehicles in challenging Indian traffic conditions. This trained model serves as the core perception component, ready for deployment on an NVIDIA Jetson Nano for real-time inference and dynamic signal control.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

The research scholar/ student must take utmost care in the use of technical abbreviations. For example, gms stands for gram meter second not grams. Grams should be abbreviated as g. Abbreviations should be listed in alphabetical orders as shown below.

AI	Artificial Intelligence
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
CPU	Central Processing Unit
CNN	Convolutional Neural Network
DBMS	Database Management System
DL	Deep Learning
GUI	Graphical User Interface
HTML	HyperText Markup Language
I/O	Input/Output
JSON	JavaScript Object Notation
LLM	Large Language Model
ML	Machine Learning
NLP	Natural Language Processing
OS	Operating System
ReLU	Rectified Linear Unit
UI	User Interface
UX	User Experience
VPS	Virtual Private Server
YOLO	You Only Look Once (Object Detection Algorithm)

NOTATION

The research scholar/student must explain the meaning of special symbols and notations used in the thesis. Define English symbols, Greek symbols and then miscellaneous symbols Some examples are listed here.

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%.

CHAPTER 1

INTRODUCTION

This is the first chapter of your thesis. Most preferably introduce your problem description and what are you discussing in the rest of the thesis in section-wise can be given here [**?**].

1.1 Objectives and Scope

Objective

The primary objective of this project phase is to create a robust, real-time object detection model specifically tailored for Indian traffic scenarios. This involves:

- Collecting and preparing a diverse dataset representative of Indian roads.
- Selecting an appropriate, high-performance object detection model (YOLOv5).
- Setting up a training environment on a GPU-enabled workstation.
- Training the model to accurately detect and classify key vehicle types (car, autorickshaw, truck, motorcycle, bus, and person).
- Evaluating the model's performance using standard computer vision metrics.

Scope

The scope of this report is strictly limited to the model creation phase. It covers the complete data-to-model pipeline, from dataset acquisition to final performance validation. This report does not cover the subsequent deployment onto the NVIDIA Jetson Nano, the integration with NVIDIA DeepStream, the development of the congestion calculation algorithm, or the hardware control of physical traffic lights via GPIO. These components are designated as the "next phase" of the project, for which this trained model is the key prerequisite.

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 project is an end-to-end intelligent system designed to replace static traffic signals. The complete system will use live camera feeds to analyze traffic, calculate congestion, and dynamically adjust signal timings to optimize traffic flow. This report focuses on the foundational first step: creating the "eyes" of the system.

We use the state-of-the-art YOLOv5s model, known for its high speed on edge computers like the Jetson Nano. To ensure the model understands the specific challenges of Indian roads, we train it on the IDD (Indian Driving Dataset), which includes unique vehicle classes and complex traffic scenes.

This training is performed on a powerful workstation equipped with an NVIDIA RTX 3050 GPU. The final output of this phase is a trained model file (`best.pt`), ready to be converted (to ONNX and TensorRT formats) and deployed on the Jetson Nano for the next phase: real-time inference and traffic control.

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.

3.3 Experimental Setup

- **Training System (Workstation):** HP Z2 Tower G9, Intel i7-12700, RTX 3050 (8GB), 32GB RAM, Ubuntu 24.04.
- **Development Environment:** VS Code, Python 3.12, PyTorch, Ultralytics YOLOv5 v7.0.
- **Dataset:** Indian Driving Dataset (IDD) – Detection split (22.8GB, 40,000 images).

3.4 Procedure, Techniques and Methodologies

The model creation followed a structured, multi-stage methodology. The high-level workflow, illustrated in Figure 3.1, ensures that the data is correctly processed and the model is trained in a repeatable and efficient manner.

The workflow begins with acquiring the raw IDD dataset, which contains images and XML annotations. This data is then put through a critical preprocessing phase, where it is reorganized and a custom script converts the annotations to the YOLO-compatible `.txt` format. Once prepared, the model is configured using a YAML file and trained on a GPU workstation. The model's performance is evaluated, and the final `best.pt` file is exported as the deployable artifact.

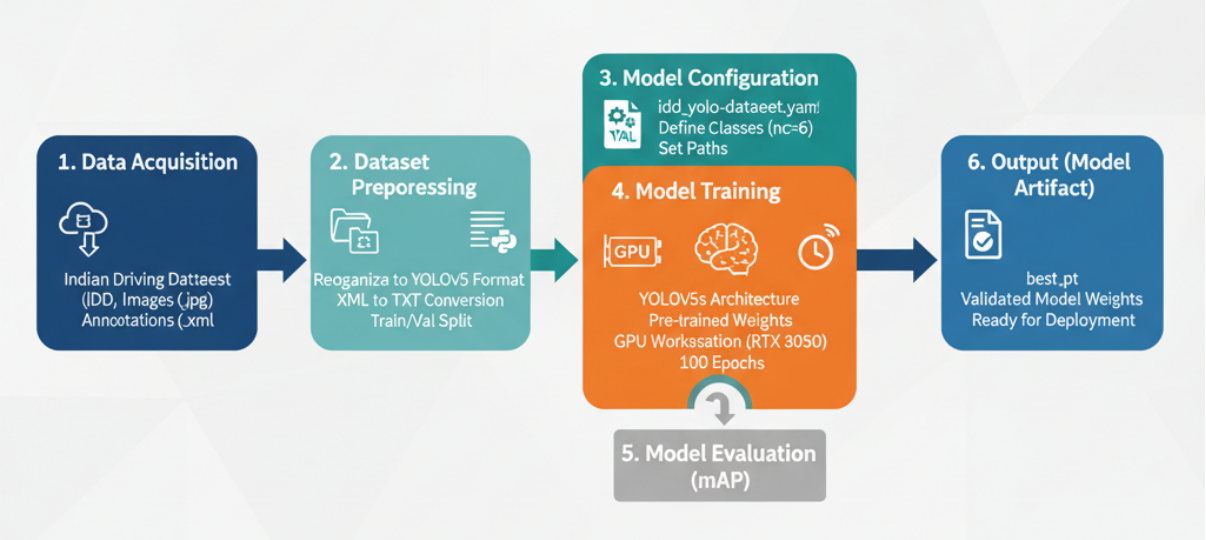


Figure 3.1: Project Workflow

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.

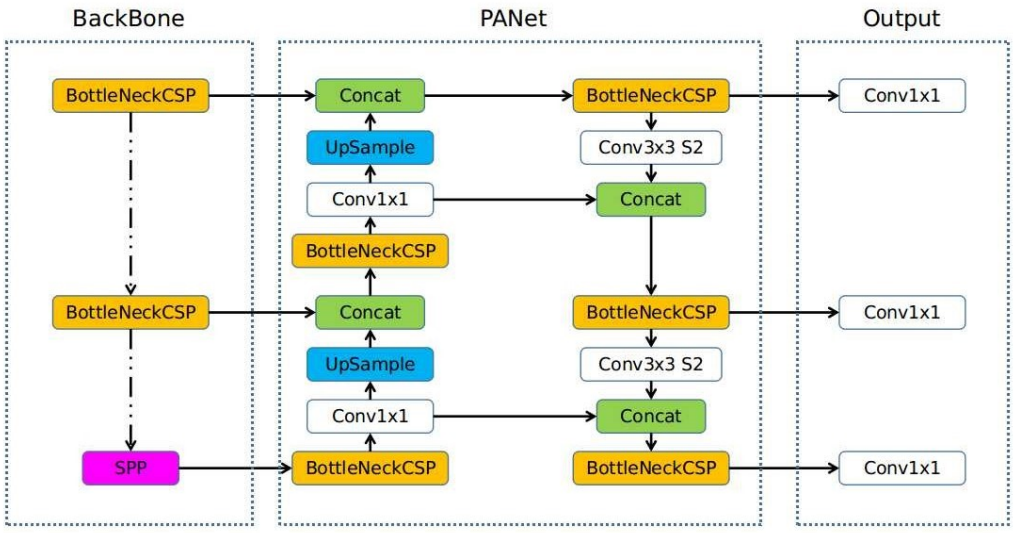


Figure 3.2: YOLO v5 architecture

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`) 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.

Class Name	Assigned ID
car	0
autorickshaw	1
truck	2
motorcycle	3
bus	4
person	5

Table 3.1: Class Mapping for YOLOv5 Training

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.

```
python train.py --img 640 --batch 8 --epochs 100 \
--data ../idd_yolo_dataset.yaml --weights yolov5s.pt \
--name IDD_yolov5s_final --cache
```

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.

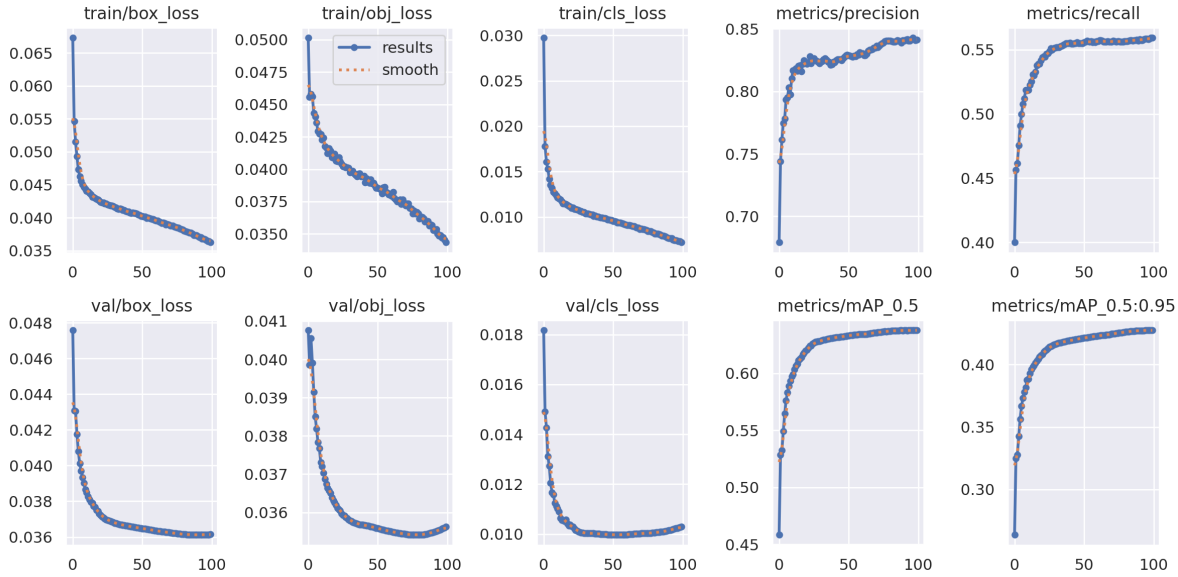


Figure 4.1: Results

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)

4.0.1 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.

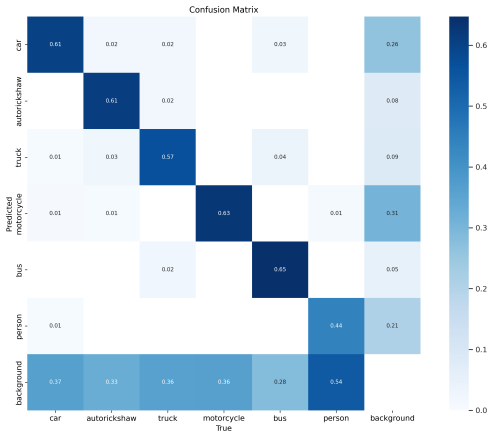


Figure 4.2: Confusion Matrix

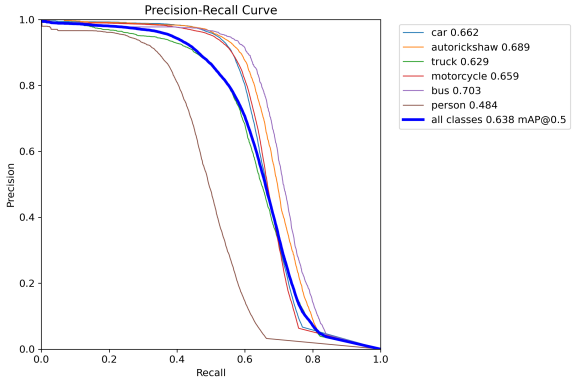


Figure 4.3: PR Curve

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.

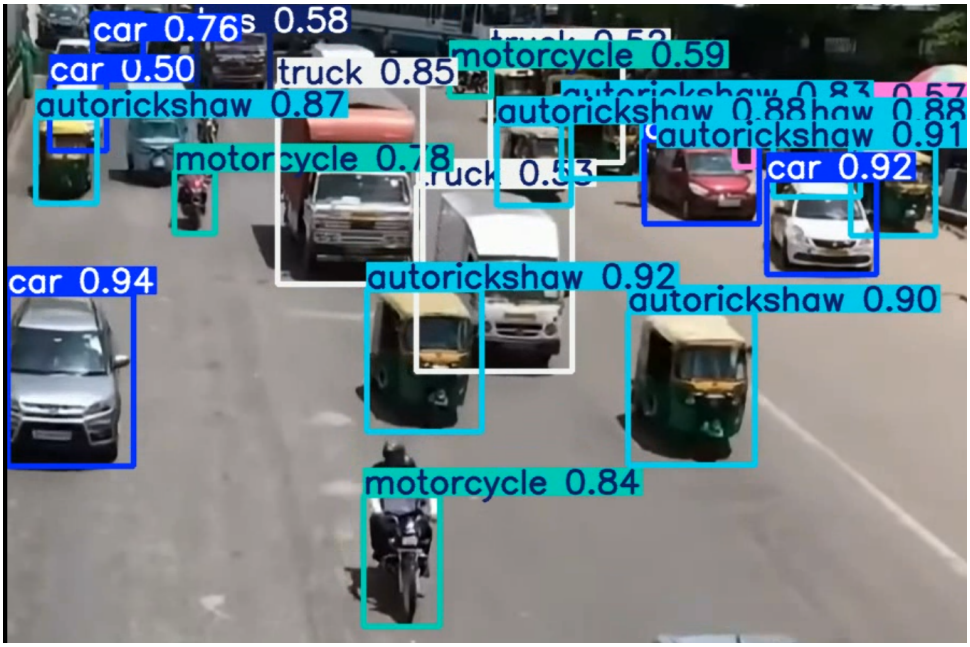


Figure 4.4: Sample Real-time Object Detection on an Indian Traffic Frame

4.0.2 Implications for Deployment

These results are highly promising for the next phase of the project. The model's high precision (0.841) is a significant strength, as it ensures that the counts sent to the congestion algorithm are reliable. This prevents "ghost" detections from causing the system to incorrectly assign green time to an empty lane.

The primary limitation, as highlighted by the moderate overall Recall (0.559), is that the system will likely *underestimate* the true number of vehicles, especially for small or heavily occluded objects like people (0.402 Recall) and motorcycles (0.589 Recall). The decision algorithm in the deployment phase must be designed to be robust to this underestimation. For example, it cannot rely on a count of zero to mean a lane is truly empty; it must incorporate a "starvation" timer to ensure that even lanes with low (or missed) detections are eventually given a chance to clear. The model is clearly validated as effective for this task and is ready for the deployment phase, where its known limitations can be managed algorithmically.

CHAPTER 5

SUMMARY AND CONCLUSION

5.1 Summary and Conclusion

This report details the successful completion of the first phase of the Intelligent Traffic Management System project: the creation of a custom, specialized object detection model. By leveraging the high-speed YOLOv5s architecture and training it on the highly relevant Indian Driving Dataset, we have produced a robust model.

The entire workflow, from data acquisition and preprocessing (converting XML to YOLO format) to training on a GPU-enabled workstation, was executed successfully. The final model achieves a strong mAP@.50 of 0.638, with excellent precision (0.841) across six key classes. While recall remains a challenge, particularly for small objects like people, the model is well-suited for its intended purpose. The resulting `best.pt` model is fully validated and serves as the foundational perception component for the next phase: deployment on the Jetson Nano for real-time congestion analysis and traffic signal control.

5.2 Scope for Future Work

With the successful creation of a validated perception model, the next part of this project involves deploying this model to build the complete, end-to-end intelligent traffic system. The scope of future work is as follows:

1. **Model Optimization and Deployment:** The trained `best.pt` model will be converted to the ONNX format and then optimized using NVIDIA TensorRT on the Jetson Nano to create a high-performance `.engine` file. This engine will be integrated into a Python application using the NVIDIA DeepStream SDK.

2. **Multi-Source Pipeline:** The DeepStream application will be configured to simultaneously process four input video streams (one for each direction of an intersection), leveraging the Jetson's hardware acceleration.

3. **Congestion Analysis Algorithm:** A Python module will be developed to analyze the real-time detection data (vehicle counts per class) from the DeepStream pipeline. This algorithm will calculate a "congestion score" for each lane, incorporating factors like weighted vehicle counts (PCU) and lane starvation timers.

4. **Hardware Integration and Control:** The decision-making algorithm will output commands that are processed by a separate thread. This thread will use the `Jetson.GPIO` library to send the physical electrical signals to the LED traffic light prototype, thus completing the "see-think-act" loop.

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.