MAHARASHTRA REMOTE SENSING APPLICATION CENTRE

INTERNSHIP REPORT

ON

MACHINE LEARNING

SUBMITTED BY

Riya Ramteke
Swayam Sahu
Priya Mankar
Shweta Rewatkar

S. B. JAIN INSTITUTE OF TECHNOLOGY,

MANAGEMENT & RESEARCH, NAGPUR

Under The Guidance Of

MR. SANJAY BALAWAR SIR

(Associate Scientist)

(Maharashtra Remote Sensing Application Centre, Nagpur)

Context/Preface

This report presents a comprehensive account of the internship completed at Maharashtra Remote Sensing Application Centre, Nagpur, over a period of **one month** in the **Machine Learning** domain. The internship provided a valuable opportunity to apply academic knowledge to practical, real-world scenarios in the field of AI-driven traffic management systems.

The primary aim of this internship was to bridge the gap between theoretical concepts and their real-world applications by engaging in hands-on project work under the guidance of experienced professionals. During this period, I was involved in the development and implementation of an intelligent traffic control solution titled **"Smart AI-Based Traffic Management System"**. This experience enabled me to acquire and enhance essential technical skills such as image processing, object detection using YOLO, traffic density estimation, and real-time signal control logic based on AI optimization techniques, including Genetic Algorithms.

This document details the organizational environment, the scope of my responsibilities, the methodologies and tools employed, and the outcomes achieved during the internship. It also reflects on the challenges encountered and the insights gained through this immersive experience in modern AI applications for smart city infrastructure.

Student Name -

Riya Ramteke

Swayam Sahu

Priya Mankar

Shweta Rewatkar

Abstract

This project presents a smart traffic signal control system designed to optimize green light durations based on real-time vehicle density and emergency vehicle detection. Leveraging **YOLOv8**, a state-of-the-art object detection model trained on the **OpenImagesV7** dataset, the system accurately identifies and counts vehicles—including cars, buses, and ambulances—from traffic images. To ensure fairness and efficiency in signal timing, a **Genetic Algorithm (GA)** is used to allocate green times dynamically across four traffic signals within a fixed cycle, based on vehicle volume. In the event an ambulance is detected, the corresponding signal is automatically given top priority by assigning it the maximum allowable green duration, overriding normal logic. The final outputs include annotated images, real-time green time calculations, and comparative bar charts that visually convey optimized signal timing. This AI-driven approach enhances traffic management by reducing congestion, improving emergency response times, and adapting to dynamic road conditions.

1. Introduction

Urban traffic congestion is a critical challenge in modern cities, often resulting in delays, pollution, and reduced emergency response efficiency. Traditional traffic signal systems rely on static timers, which do not adapt to real-time traffic conditions or emergency situations. To address this, our project presents a smart traffic signal management system that dynamically adjusts signal green times using **AI-based vehicle detection** and **genetic algorithms**, with priority handling for **emergency vehicles like ambulances**.

This system leverages the **YOLOv8 object detection model** trained on the OpenImagesV7 dataset to detect and count various types of vehicles—including cars, buses, and vans—from real-time traffic images. It also identifies ambulances with high precision using class-based logic and keyword filtering.

After detecting the vehicle count from four different traffic signal inputs, a **Genetic Algorithm (GA)** is used to optimize the distribution of green time across all signals within a fixed cycle duration (e.g., 120 seconds). The algorithm ensures that signals with higher vehicle density are allotted more green time. Additionally, any signal with an ambulance detected is immediately given maximum priority by assigning it the highest allowable green time.

The final output includes:

- Annotated images showing detected vehicles and emergency vehicles,
- Calculated vehicle counts,
- Optimized green time durations per signal,
- A bar graph illustrating the green time distribution.

By integrating **real-time object detection** with **evolutionary optimization**, this project offers a scalable and intelligent approach to traffic signal control that not only improves traffic flow but also enhances emergency vehicle clearance.

2. Internship Objectives

The primary objective of this internship is to gain hands-on experience in applying artificial intelligence and machine learning techniques to real-world problems, with a focus on intelligent traffic management systems. The project aims to enhance both technical knowledge and practical skills in the domains of computer vision, deep learning, and optimization algorithms.

Specific Objectives:

1. **To study and implement real-time vehicle detection:**
   using the YOLOv8 object detection model trained on the OpenImagesV7 dataset.

2. **To design and develop a system that can count vehicles:**
   from traffic images and identify emergency vehicles like ambulances with high accuracy.

3. **To apply Genetic Algorithms (GA):**
   for optimizing green signal durations based on real-time vehicle density.

4. **To implement emergency vehicle prioritization logic:**
   that dynamically adjusts green times to facilitate faster ambulance movement.

5. **To visualize and evaluate the results:**
   through annotated images and graphical representations such as bar charts.

6. **To enhance understanding of AI model integration:**
   with optimization techniques and improve skills in Python, OpenCV, and Matplotlib.

7. **To contribute to the development of smart city solutions:**
   by creating a scalable and adaptive traffic control prototype.

3. Roles and Responsibilities

During the course of this internship, I undertook several technical and research-oriented tasks aimed at designing, developing, and testing an intelligent traffic signal control system. The following roles and responsibilities were carried out:

- **Developed and implemented YOLOv8 object detection** to identify and count vehicles from real-time traffic images, including the detection of emergency vehicles like ambulances using label and class-based logic.

- **Designed a priority-based control mechanism** to assign maximum green time to signals where ambulances were detected, ensuring faster emergency vehicle clearance.

- **Applied a Genetic Algorithm (GA)** to optimize traffic signal green durations based on detected vehicle density, while maintaining constraints like total cycle time and min-max green limits.

- **Processed and annotated images** using OpenCV to visualize detections with bounding boxes and labels for each vehicle class.

- **Created bar charts and visual summaries** using Matplotlib to display green time distribution for all traffic signals.

- **Wrote and debugged Python scripts** to integrate object detection, optimization, and result visualization into a seamless workflow.

- **Maintained documentation** of the entire development process and contributed to the final project report and result analysis.

4.Key Skills Development

Throughout the course of this internship, a wide range of technical, software, and professional skills were developed. These skills reflect both the theoretical understanding and practical application of AI and optimization techniques in real-world traffic systems.

1. Technical Skills

- Applied **YOLOv8** for real-time vehicle and ambulance detection.
- Implemented **Genetic Algorithms** for traffic signal optimization.
- Worked with **OpenCV** for image processing and annotation.
- Visualized results using **Matplotlib** for better interpretation.

2. Software Proficiency

- Proficient in **Python**, **Google Colab**, and libraries like **Ultralytics**, **DEAP**, **OpenCV**, and **Matplotlib**.
- Hands-on experience using pretrained models and cloud-based development tools.

3. Professional Skills

- Strengthened **problem-solving** and **critical thinking** through real-world AI applications.
- Improved **team collaboration**, **time management**, and **technical documentation**.
- Gained confidence in presenting and communicating complex concepts clearly.

5. Project Undertaken: AI-Based Traffic Signal Optimization with Emergency Vehicle Priority.

5.1. Introduction of Project

Traffic congestion is one of the most pressing issues in rapidly growing urban areas. Traditional traffic control systems use static timing plans, which are often unable to respond to fluctuating vehicle density or provide timely clearance for emergency vehicles like ambulances. This causes delays, fuel wastage, and critical risk to human life in emergencies.

The primary aim of this project is to develop a smart, AI-powered traffic management system that can:

- Automatically detect and count vehicles at multiple intersections.
- Identify emergency vehicles like ambulances.
- Allocate optimized green signal durations based on traffic density.
- Grant **priority clearance to ambulances** by overriding normal logic.

By integrating **YOLOv8 object detection** with **Genetic Algorithm-based signal timing optimization**, this system presents a scalable and intelligent approach to urban traffic control.

5.2. Project Aim:

This project aims to develop a **smart and responsive traffic signal control system** that not only adapts to the **real-time traffic load** at each signal but also **prioritizes emergency vehicles**, ensuring quick and unhindered passage.

By integrating **deep learning for vehicle detection** and **Genetic Algorithms for traffic signal optimization**, this system simulates a modern, intelligent traffic control approach that aligns with the goals of **smart city development** and **public safety enhancement**.

5.3. Impact and Innovation:

What sets this project apart is the **combination of object detection and optimization**, making it both technically robust and practically impactful. Instead of manually adjusting signal times or relying on timers, this system uses **real-time visual data** and **intelligent algorithms** to make traffic decisions. It represents a major step forward in automating and modernizing urban traffic systems.

5.4. Methodology

The methodology adopted for this project is a step-by-step integration of deep learning-based object detection with evolutionary optimization techniques to create a smart and adaptive traffic control system. The core workflow includes **image acquisition**, **vehicle detection**, **ambulance identification**, **green time optimization using a Genetic Algorithm**, and **output visualization**. Each step is detailed below:

Step 1: Data Input and Image Acquisition

- The system starts by prompting the user to **upload exactly four traffic images**.
- These images represent **four different traffic signal intersections**, each depicting the traffic scenario at a specific location.
- Images are uploaded manually for testing purposes using google.colab.files.upload().
- In real-world deployment, these images could be captured in real-time using **CCTV cameras** at intersections.

Step 2: Object Detection Using YOLOv8

- The uploaded images are passed into a pretrained **YOLOv8 (You Only Look Once - Version 8)** model.
- The model used is trained on the **OpenImagesV7** dataset, which contains over 600 classes including various vehicles and emergency services.
- YOLOv8 processes each image and returns:
    - **Bounding boxes** around detected objects.
    - **Class IDs and class labels** (e.g., car, truck, ambulance, bus).
    - **Confidence scores** for each prediction.
- This step ensures fast and efficient multi-object detection across varying environments and lighting conditions.

Step 3: Ambulance Detection Logic

- Once the detections are made, the system applies logic to specifically identify ambulances:
    - **Primary check**: If the detected class ID equals that of an ambulance (e.g., class ID = 6).
    - **Secondary check**: If the detected label contains the word **"ambulance"** (case-insensitive).
    - **Tertiary check**: For ambiguous cases like **vans** (which could visually resemble ambulances), a confidence threshold (>0.7) is applied to accept or reject the detection.
- If an ambulance is detected at any signal, it is flagged for **priority treatment**.

Step 4: Vehicle Counting

- For each image, the total number of detected vehicles (cars, buses, trucks, motorcycles, etc.) is counted using:

    - Class ID matching against a predefined vehicle list.
    - Filtering out detections with low confidence (e.g., below 0.2).
- The result is a **vehicle count per signal**, which reflects the **traffic load** at that intersection.

Step 5: Green Time Optimization Using Genetic Algorithm

A **Genetic Algorithm (GA)** is used to optimize green time allocation for each of the 4 signals. This algorithm mimics natural selection to find an optimal or near-optimal solution for complex problems.

a. Problem Setup:

- Total cycle time = 120 seconds.
- Minimum green time = 10 seconds.
- Maximum green time = 60 seconds.
- The algorithm searches for the best combination of green times [t1, t2, t3, t4] that satisfy:
    - All times $\geq$ 10s and $\leq$ 60s.
    - Sum of all times = 120s.
    - Distribution should match the ratio of vehicle counts (i.e., more traffic $\rightarrow$ more time).

b. Fitness Function:

- Calculates the **squared deviation** between ideal time ratios (based on traffic) and actual time allocation.
- Penalizes individuals violating constraints (e.g., total time $\neq$ 120s or invalid range).
- Objective: **minimize total deviation**.

c. GA Parameters:

- Population Size: 50 individuals.
- Number of Generations: 100.
- Crossover: cxBlend, with blending factor $\alpha = 0.5$.
- Mutation: mutGaussian, with standard deviation = 5.
- Selection: Tournament Selection with tournament size = 3.

d. Output:

- After evolution, the best solution (individual with minimum fitness score) is selected.
- This gives the optimized green time for each signal.

Step 6: Ambulance Priority Enforcement

- If any signal is flagged with **ambulance presence**, it is assigned the **maximum green time** (60s).

- The remaining green time (total_time_left = 120 - 60) is then proportionally distributed among the other signals using the GA-generated solution.
- This ensures **emergency clearance** while still accounting for traffic load at the remaining signals.

Step 7: Output Generation and Visualization

- **Annotated Images**:
  - Bounding boxes are drawn on vehicles.
  - Labels such as "Car", "Bus", or "AMBULANCE" are overlaid.
  - Total vehicle count and ambulance status are displayed on each image.
- **Bar Graph Visualization**:
  - A bar graph is created using **Matplotlib** showing green time for each signal.
  - Signals with ambulance priority are highlighted using different colors or emojis (e.g., 🚑).
- **Console Output**:
  - Detailed printout of each signal's:
    - Vehicle count
    - Ambulance flag
    - Final green signal duration (in seconds).

5.5. Components Used

- **Software Tools**:
  - Python (main programming language)
  - Google Colab (for execution and testing)
  - Ultralytics YOLOv8 model
  - OpenCV (for image processing)
  - Matplotlib (for data visualization)
  - DEAP library (for implementing Genetic Algorithm)

- **Hardware Requirements**:
  - Laptop/PC with internet
  - Camera/traffic feed (simulated with images)

5.6. Working of the Project

The project is designed to simulate and demonstrate how an intelligent traffic signal control system can work using artificial intelligence (AI) and genetic optimization techniques. The system takes four traffic signal inputs (images), detects the number of vehicles at each signal, identifies if an ambulance is present, and dynamically allocates green signal durations accordingly.

◆ Step-by-Step Workflow

1. Image Input and Upload

- The user uploads **four traffic images**, each representing the view from a different traffic signal.
- These images simulate real-time input, just like traffic cameras would feed in a live environment.

2. Vehicle Detection Using YOLOv8

- Each image is passed through the **YOLOv8** object detection model, pretrained on the **OpenImagesV7** dataset.
- YOLOv8 detects and classifies vehicles such as:
  - Cars
  - Buses
  - Trucks
  - Motorcycles
  - Vans
  - Ambulances
- Bounding boxes and confidence scores are returned for every object detected.

3. Ambulance Identification

- Special logic is applied to detect **ambulances**, which may appear under specific class IDs or as vans labeled "ambulance."
- The system flags any signal containing an ambulance and gives it top priority in the green time allocation.

4. Vehicle Counting Per Signal

- Vehicles are counted individually for each signal image based on valid detection results (confidence > 0.2).
- The system excludes low-confidence detections to improve accuracy.

5. Green Time Optimization Using Genetic Algorithm

- A **Genetic Algorithm (GA)** is used to optimize how a fixed total green signal cycle (e.g., 120 seconds) is distributed across the 4 signals.
- The algorithm uses:
  - **Fitness Function**: Penalizes deviation from ideal time proportions based on vehicle counts.
  - **Constraints**: Green time must stay within a min-max range (10–60 seconds per signal).
- The GA simulates evolution: individuals (solutions) are created, evaluated, crossed, and mutated until the best signal timing is found.

6. Emergency Signal Adjustment

- If an **ambulance is detected** at any signal:
  - That signal is **instantly assigned the maximum green time (e.g., 60s)**.
  - Remaining time is rebalanced among other signals using proportional reduction to maintain the 120s total cycle time.

7. Visualization and Output

- **Annotated Images**: Each image is displayed with bounding boxes around detected vehicles and labels showing counts and ambulance presence.
- **Graphical Representation**: A **bar chart** shows the optimized green signal times for all four signals.
- **Printed Summary**:
  - Vehicle counts per signal
  - Final green time allocation
  - Emergency signal priority status

5.7. Source Code

```
# Step 1: Install required libraries
!pip install ultralytics opencv-python matplotlib deap –quiet

# Step 2: Import libraries
from ultralytics import YOLO
import cv2
import matplotlib.pyplot as plt
from google.colab import files
import numpy as np
from deap import base, creator, tools, algorithms
import random

# Step 3: Upload 4 traffic images
print("Upload exactly 4 images (each showing a traffic signal):")
uploaded = files.upload()
image_paths = list(uploaded.keys())[:4]
if len(image_paths) != 4:
    raise ValueError("Please upload exactly 4 images.")

# Step 4: Load a more accurate YOLOv8 model (Open Images V7)
model = YOLO("yolov8s-oiv7.pt")  # pretrained on OpenImagesV7

# Define class IDs
vehicle_class_ids = [90, 73, 558, 342, 550]  # Car, Bus, Truck, Motorcycle, Train
ambulance_class_id = 6
van_class_id = 564  # fuzzy logic previously used

vehicle_counts = []
ambulance_present = []
images = []

# Step 5: Detection
for img_path in image_paths:
    print(f"\nAnalyzing Image: {img_path}")
    img = cv2.imread(img_path)
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    results = model(img_rgb, imgsz=640)
    res = results[0]
    boxes = res.boxes

    xyxy = boxes.xyxy.cpu().numpy()
    confs = boxes.conf.cpu().numpy()
    cls_ids = boxes.cls.cpu().numpy().astype(int)
```

```
    vehicle_count = 0
    ambulance_flag = False

    for (x1, y1, x2, y2), conf, cls in zip(xyxy, confs, cls_ids):
        label = model.names[cls].lower()
        print(f"  Detected: {label} (ID {cls}) with confidence {conf:.2f}")

        if conf < 0.2:
            continue  # Slightly lowered threshold to catch motorcycles

        is_ambulance = (
            cls == ambulance_class_id or
            'ambulance' in label or
            (cls == van_class_id and conf > 0.7)
        )

        if is_ambulance:
            ambulance_flag = True
            vehicle_count += 1  # Count ambulance as a vehicle
            cv2.rectangle(img_rgb, (int(x1), int(y1)), (int(x2), int(y2)), (255, 0, 0), 3)
            cv2.putText(img_rgb, "AMBULANCE",
                        (int(x1), int(y1) - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)
        elif cls in vehicle_class_ids:
            vehicle_count += 1
            cv2.rectangle(img_rgb, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255, 0), 2)
            cv2.putText(img_rgb, label.capitalize(), (int(x1), int(y1) - 5),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0), 2)

    vehicle_counts.append(vehicle_count)
    ambulance_present.append(ambulance_flag)

    # Annotate vehicle count
    cv2.putText(img_rgb, f"Vehicles: {vehicle_count}",
                (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1,
                (255, 0, 0) if ambulance_flag else (255, 255, 0), 2)

    images.append(img_rgb)

# Step 6: Genetic Algorithm setup
CYCLE_TIME, MIN_GREEN, MAX_GREEN, N_SIGNALS = 120, 10, 60, 4

creator.create("FitnessMax", base.Fitness, weights=(1.0,))
creator.create("Individual", list, fitness=creator.FitnessMax)

def evaluate(ind):
    total = sum(ind)
```

```
    if total == 0:
        return -float('inf'),
    norm = [t * CYCLE_TIME / total for t in ind]
    if any(t < MIN_GREEN or t > MAX_GREEN for t in norm):
        return -float('inf'),
    total_vehicles = sum(vehicle_counts)
    if total_vehicles == 0:
        return -float('inf'),
    ideal = [v / total_vehicles for v in vehicle_counts]
    actual = [t / CYCLE_TIME for t in norm]
    dev = sum((i - a) ** 2 for i, a in zip(ideal, actual))
    return -dev,

def init_individual():
    vals = [random.uniform(MIN_GREEN, MAX_GREEN) for _ in range(N_SIGNALS)]
    total = sum(vals)
    return creator.Individual([v * CYCLE_TIME / total for v in vals])

toolbox = base.Toolbox()
toolbox.register("individual", tools.initIterate, creator.Individual, init_individual)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
toolbox.register("evaluate", evaluate)
toolbox.register("mate", tools.cxBlend, alpha=0.5)
toolbox.register("mutate", tools.mutGaussian, mu=0, sigma=5, indpb=0.2)
toolbox.register("select", tools.selTournament, tournsize=3)

pop = toolbox.population(n=50)
for _ in range(100):
    offspring = algorithms.varAnd(pop, toolbox, cxpb=0.7, mutpb=0.3)
    fits = list(map(toolbox.evaluate, offspring))
    for ind, fit in zip(offspring, fits):
        ind.fitness.values = fit
    pop = toolbox.select(offspring, k=len(pop))

best = tools.selBest(pop, k=1)[0]
green_times = [round(t * CYCLE_TIME / sum(best), 1) for t in best]

# Step 7: Prioritize ambulance signals
for i in range(N_SIGNALS):
    if ambulance_present[i]:
        green_times[i] = MAX_GREEN

# Normalize if total time exceeds cycle
total_time = sum(green_times)
if total_time > CYCLE_TIME:
    extra = total_time - CYCLE_TIME
```

```
    other_idxs = [i for i in range(N_SIGNALS) if not ambulance_present[i]]
    total_other = sum(green_times[i] for i in other_idxs)
    for i in other_idxs:
        green_times[i] -= extra * (green_times[i] / total_other)

# Step 8: Show annotated images
plt.figure(figsize=(12, 8))
for i, (img, vc, gt, amb) in enumerate(zip(images, vehicle_counts, green_times,
ambulance_present)):
    plt.subplot(2, 2, i + 1)
    plt.imshow(img)
    plt.axis('off')
    title = f"Signal {i+1}: {vc} Vehicles\nGreen Time: {round(gt)}s"
    if amb:
        title += " 🚑 PRIORITY"
    plt.title(title)
plt.tight_layout()
plt.show()

# Step 9: Print results
print("\nVehicle Counts and Green Times (w/ Ambulance Priority):")
for i in range(4):
    priority = " " if ambulance_present[i] else ""
    print(f"Signal {i+1}: {vehicle_counts[i]} vehicles → Green Time:
{round(green_times[i])}s{priority}")

# Step 10: Bar chart of green times
plt.figure(figsize=(7, 5))
labels = [f"S{i+1}" for i in range(4)]
colors = ['#4CAF50', '#2196F3', '#FFC107', '#FF5722']
plt.bar(labels, green_times, color=colors, edgecolor='black')
plt.ylabel("Green Time (seconds)")
plt.title("Optimized Signal Green Times")
for i, v in enumerate(green_times):
    plt.text(i, v + 1, f"{round(v)}s", ha='center', fontweight='bold')
plt.ylim(0, MAX_GREEN + 10)
plt.tight_layout()
plt.show()
```

5.8. Applications

1. Smart Traffic Signal Control

The primary application of this system is in **automated traffic signal control**, where signal green times are adjusted dynamically based on real-time traffic density. This improves overall traffic flow, reduces wait times, and helps in decongesting heavily loaded junctions.

- **Benefit**: Replaces static timer systems with intelligent, adaptive logic.
- **Use Case**: Busy intersections in metropolitan cities like Mumbai, Delhi, or Bengaluru.

2. Emergency Vehicle Prioritization

This system is especially beneficial for **ambulances**, ensuring they receive maximum green time at signals to avoid delays. Early detection and automated prioritization reduce emergency response times and can help save lives.

- **Benefit**: Ensures ambulances are not delayed by red lights.
- **Use Case**: Hospital corridors, accident-prone areas, and ambulance routes.

3. Smart City Integration

This project aligns perfectly with the vision of **Smart Cities**, which require intelligent infrastructure for efficient operation. It can be integrated with **IoT**, **cloud-based monitoring**, and **central traffic control rooms**.

- **Benefit**: Scalable and remotely manageable via central systems.
- **Use Case**: Deployment in Smart City projects under government initiatives (e.g., India's Smart Cities Mission).

4. Public Transport Signal Priority

Besides emergency vehicles, this system can be extended to recognize **public transport** such as buses or trams and give them signal priority, reducing delays and improving the reliability of public transport.

- **Benefit**: Supports sustainable and efficient urban mobility.
- **Use Case**: Bus rapid transit systems (BRTS) or tram corridors.

5. Real-Time Traffic Monitoring and Reporting

Provides live vehicle counts and congestion data for traffic analysis and planning.

- **Benefit**: Acts as a traffic analytics tool for city planning authorities.
- **Use Case**: Real-time dashboards for traffic control centers.

5.9. Advantages

1. Real-Time Traffic Adaptation

The system dynamically adjusts green signal durations based on actual vehicle density captured in real-time, improving traffic flow and reducing idle time.

- **Benefit**: Decreases wait times and fuel wastage.
- **Impact**: Better congestion handling during peak hours.

2. Emergency Vehicle Priority

Ambulances are detected and prioritized instantly, ensuring the traffic signal grants them immediate passage by assigning maximum green time.

- **Benefit**: Reduces delay in emergency response.
- **Impact**: Potential to save lives during critical situations.

3. AI-Based Decision Making

Using YOLOv8 and Genetic Algorithms ensures decisions are data-driven, optimized, and more accurate than manual or timer-based systems.

- **Benefit**: Enhances efficiency and automation.
- **Impact**: Consistent, unbiased traffic control.

4. Scalability and Flexibility

The system can be easily scaled to multiple intersections and integrated with live CCTV feeds for broader city-wide implementation.

- **Benefit**: Suitable for Smart City deployment.
- **Impact**: Cost-effective long-term solution.

5. Eco-Friendly Operations

By reducing idle times and unnecessary stoppages, the system contributes to lower vehicle emissions and improved air quality.

- **Benefit**: Supports environmental sustainability.
- **Impact**: Helps in pollution control initiatives.

5.10. Disadvantages

1. Dependence on Image Quality

The accuracy of detection depends heavily on the clarity and lighting of the input images. Poor-quality feeds may lead to false or missed detections.

- **Impact**: May fail in foggy, rainy, or low-light conditions.

2. Limited Ambulance Detection Accuracy

If the ambulance is partially occluded, misclassified, or not labeled correctly in the dataset, it may not be identified.

- **Impact**: Ambulance might not receive priority as expected.

3. Fixed Signal Count

The current system is designed for 4 signals. Expanding it to more junctions would require additional customization and testing.

- **Impact**: Needs adaptation for large-scale deployment.

4. No Real-Time Video Processing Yet

The project currently works on static images, not live video feeds. Transitioning to continuous real-time processing requires higher computational resources.

- **Impact**: Limits real-world deployment in its current state.

5. Processing Speed Limitations

Running YOLOv8 and Genetic Algorithms in Google Colab or low-end systems may introduce latency.

- **Impact**: Delays in decision-making if not optimized for edge computing.

5.11. Future Scope

This project lays the foundation for intelligent, responsive traffic systems and has great potential for expansion and enhancement. Future developments may include:

- **Live Video Integration**: Shift from static images to real-time video feeds using CCTV or drone surveillance for continuous monitoring and optimization.
- **Edge Computing Deployment**: Implement the system on edge devices (like NVIDIA Jetson Nano) for on-site, low-latency decision-making.
- **AI Model Retraining**: Use locally collected data to retrain YOLOv8 for improved accuracy on region-specific vehicle types (e.g., rickshaws, e-carts).
- **Public Transport Priority**: Extend ambulance priority logic to prioritize public buses and school vehicles for smarter urban mobility.

5.12. Challenges Faced

Despite successful implementation, several challenges arose during the project:

- **Model Accuracy in Variable Conditions**: YOLOv8 struggled with partial occlusion, low lighting, and crowded scenes, affecting detection reliability.
- **Ambulance Identification**: Ambiguity in identifying ambulances when misclassified or visually similar to vans required multiple conditional checks.
- **Resource Limitations**: Processing heavy models and algorithms in Google Colab sometimes led to delays or session timeouts.
- **Image Constraints**: Relying on static images rather than live feeds limited the real-world applicability during testing.
- **Genetic Algorithm Tuning**: Achieving a balanced solution that honored both green time constraints and vehicle priority required fine-tuning of mutation/crossover rates.

5.13. Learning Outcomes

This internship provided rich exposure to cutting-edge technologies and real-world problem solving. Major learning outcomes include:

- **Technical Mastery**:
  - Developed skills in **YOLOv8**, **OpenCV**, **Matplotlib**, and **Genetic Algorithms**.
  - Understood the workflow of object detection and optimization-based decision systems.

- **Problem Solving**:
  - Tackled real-world challenges like congestion and emergency prioritization using AI techniques.

- **Practical Experience**:
  - Bridged theory and practice by applying machine learning in a civic context.
  - Gained experience working with real data and converting results into actionable visual insights.

- **Professional Development**:
  - Improved skills in **project planning**, **report writing**, **technical documentation**, and **collaborative work**.

5.14. Conclusion

The project successfully demonstrated a smart AI-based traffic signal system capable of optimizing green time distribution based on real-time traffic analysis while ensuring priority clearance for emergency vehicles. By combining **YOLOv8 object detection** with a **Genetic Algorithm**, the system not only improved traffic efficiency but also addressed critical aspects of urban safety and emergency response.

This solution serves as a promising step toward building **intelligent transportation systems** for smart cities. With further development—such as live feed integration and hardware optimization—it holds great potential for deployment in real-world environments to make urban mobility faster, safer, and more intelligent.