The Hashemite University
Faculty of Prince Al-Hussein Bin Abdallah II for Information Technology
Information Technology Department

Smart Traffic Management System

A project submitted in partial fulfilment of the requirements for the B.Sc. Degree in Data science and Artificial Intelligence.

By

Shahed Mowaffaq Al-Zu’bi (2137097)   Awad Abdo Joudeh (2132401)

Omama Musa Rawashdeh (2137771)   Momen Yasin Al-Khalilah (2130462)

Supervised by

Dr. Eman Sharif Omar

Committee Member Name

Dr. Ma’moon Mohammed Obiedat
Dr. Hind Melhem

Spring 2025

CERTIFICATE

- It is hereby certified that the project titled **Smart Traffic Management System** submitted by undersigned, in partial fulfilment of the award of the degree of “bachelor's in data science and Artificial Intelligence” embodies original work done by them under my supervision.
- All the analysis, design and system development have been accomplished by the undersigned. Moreover, this project has not been submitted to any other college or university.

Shahed Mowaffaq Al-Zu’bi (2137097)

Awad Abdo Joudeh (2132401)

Omama Musa Rawashdeh (2137771)

Momen Yasin Al-Khalilah (2130462)

ABSTRACT

Traffic congestion is a significant issue in cities worldwide, resulting in billions of dollars lost due to reduced productivity and wasted fuel. It also contributes to higher levels of air pollution and increased stress for commuters. Traditional traffic light systems typically rely on fixed timing schedules, which can be inefficient, especially during peak traffic periods, often leading to further congestion.

To address this, Intelligent Traffic Lights (ITLs) leverage Artificial Intelligence (AI) to monitor real-time traffic conditions and dynamically adjust the timing of traffic signals. This approach helps alleviate congestion, optimize traffic flow, and reduce emissions.

One key AI technology that can aid in this solution is Computer Vision (CV). By using CV algorithms to detect vehicles and assess their density at various intersections, traffic light timing can be adjusted in real time, minimizing traffic jams and improving overall road efficiency.

The YOLOv11 algorithm, one of the most accurate and fastest real-time object detection models in the field of computer vision, will serve as the foundation for our project. By using YOLOv11 to detect vehicles on the road, we can adjust traffic light timings based on vehicle density and traffic patterns.

Looking ahead, our project envisions a more integrated system—where a centralized control unit utilizes real-time traffic data for continuous optimization of traffic signals. This anticipatory approach will enhance the performance of ITLs, aligning with the evolving needs of modern cities.

ACKNOWLEDGEMENTS

The successful completion of this project would not have been possible without the unwavering support, guidance, and inspiration of many individuals.

To our families, who stood by us during long nights and challenging moments, your unconditional love and support have been the backbone of our perseverance. Your belief in our abilities has been a source of strength that kept us moving forward.

We are also immensely grateful to our peers and teammates, whose collaboration and dedication enriched this project. Working alongside such talented and committed individuals has been a truly rewarding experience.

Finally, and for the most, we extend our deepest gratitude to our project supervisor, whose invaluable insights, patience, and expertise guided us through every stage of this journey.

This project stands as a testament to the power of teamwork, perseverance, and shared dreams. We thank each of them who played a part in making this accomplishment possible.

TABLE OF CONTENTS

ABBREVIATIONS

ITLs: Intelligent Traffic Lights.
STMS: Smart Traffic Management System.
MFA: Multi Factor Authentication.
MTBF: Mean Time Between Failures.
TIMMS: Transportation Intelligent Mobility Management System.
TMC: Transportation Management Centre.
RTA: Road and Transportation Authority.
ATSC: Adaptive Traffic Signal Control.
GAM: Greater Amman Municipality.
ATSAC: Automated Traffic Surveillance and Control.
TOPIS: Transport Operation and Information Service.
API: Application Programming Interface.
GPS: Global Positioning System.
GDPR: General Data Protection Regulation.
PCA: Principal Component Analysis.
DBSCAN: Density-Based Spatial Clustering of Applications with Noise.
SMOTE: Synthetic Minority Over-sampling Technique.
ADASYN: Adaptive Synthetic Sampling.
R-CNN: Region-based Convolutional Neural Networks.
RFE: Recursive Feature Elimination.
LASSO: Least Absolute Shrinkage and Selection Operator.

LIST OF FIGURES

LIST OF TABLES

Chapter 1: Introduction

Traffic congestion in Jordan’s urban centers, particularly Amman, poses significant challenges due to rising vehicle numbers and outdated traffic management systems. Fixed-timing traffic lights fail to adapt to real-time conditions, leading to delays, increased fuel consumption, and higher pollution levels. The Smart Traffic Management System (STMS) leverages AI, specifically YOLOv11 for computer vision and YAMNet for audio classification, to enable real-time traffic monitoring, adaptive signal control, and emergency vehicle prioritization. This project aims to optimize traffic flow, enhance road safety, and reduce environmental impacts in Jordan’s urban areas.

1.1 Overview

Urban population growth has intensified traffic congestion, causing delays, fuel wastage, and environmental degradation. Traditional traffic light systems, reliant on fixed schedules, exacerbate congestion during peak hours. STMS addresses these issues by integrating real-time data from cameras and microphones to dynamically adjust traffic signals. Using YOLOv11, the system detects vehicles and traffic lights, while YAMNet identifies emergency sirens, enabling priority for ambulances. This approach enhances safety, reduces emissions, and improves urban mobility in Jordan.

1.2 Project Motivation

Traffic congestion disrupts daily life, causing delays, increasing fuel costs, and contributing to air pollution. In Jordan, reliance on fixed-timing traffic lights leads to inefficiencies, particularly during variable traffic conditions. This project is motivated by the need to address these challenges using AI and computer vision, offering a scalable solution to improve travel times, reduce environmental impact, and enhance quality of life for Jordanian citizens

1.3 Problem Statement

Traffic congestion in Jordan's urban areas, driven by increased vehicle usage and inadequate infrastructure, results in delays, higher fuel consumption, and pollution. Current fixed-timing traffic systems fail to adapt to real-time conditions, exacerbating congestion. STMS aims to reduce congestion, improve travel times, enhance road safety, and optimize resource usage by leveraging AI for adaptive traffic control, with scalability for integration into Jordan's infrastructure.

1.4 Project Aim and Objectives

Aim:
To implement a real-time traffic management system using computer vision, machine learning and audio analysis to optimize traffic flow, reduce congestion, and improve road safety in Jordan's urban areas.

Objectives:

- Develop real-time vehicle detection and classification for cars, buses, trucks, motorcycles, and ambulances.
- Monitor traffic flow and reduce congestion using density analysis.
- Enable dynamic traffic signal control based on real-time conditions.
- Prioritize emergency vehicles using visual and audio detection.
- Reduce environmental impact by minimizing idle times.
- Ensure scalability and integration with existing infrastructure.
- Analyze long-term traffic data for urban planning.

1.5 Project Expected Output

- **Real-time Data Collection**: Gather vehicle counts, traffic density, and siren detection from cameras and microphones.
- **Traffic Flow Analysis**: Analyze density and congestion patterns to optimize signal timings.
- **Emergency Prioritization**: Detect ambulances and sirens to adjust signals for faster response.
- **Environmental Impact**: Reduce emissions through efficient traffic flow.
- **Scalable System**: Deploy a web-based interface for real-time monitoring and future IoT integration.

1.6 Project Schedule

PHASES	TIME
Planning: Define scope, objectives, and requirements	2 Weeks
Research: Review of smart traffic systems and tools	2 Weeks
Data Collection: Gather and preprocess traffic data	5 Weeks
Implementation: Develop models and integrate components	3 Weeks expected
Testing: Conduct unit, integration, and performance tests	2 Weeks expected
Conclusion: Prepare report and suggest enhancements	2 Weeks expected

1.7 Report organization

In this report we organized the project into seven chapters, each one addressing a different aspect of the project. Below is a structured overview of the document to guide you through its contents:

- Chapter 1: Introduction.
This chapter introduces our project and what it will cover, including an overview, motivation, problem statement, aims and objectives, expected outcomes, and report organization.
- Chapter 2: Literature Review.
In chapter 2 we provide an in-depth review of relevant literature and existing studies or applied projects in the fields of Smart traffic management (AI-based), computer vision for traffic analysis, and dynamic traffic control systems. This chapter highlights key points and concepts, methodologies, and technological advancements, establishing the foundational knowledge for our project.

- Chapter 3: Requirement Engineering and Analysis.
This chapter outlines the requirements for the proposed system, covering both functional and non-functional requirements. It also details the system's specifications, including data sources, hardware and software requirements, and technical constraints necessary for successful implementation.
It explores the data preparation process, including collection methods, preprocessing techniques, and ethical considerations. Additionally, it describes the features of engineering and the impact of phenomena like phase sanitization and the Doppler effect.
- Chapter 4: Model Development and Architectural Design.
Chapter 4 describes the architectural design, data flow diagrams, and detailed explanations of the AI techniques we need to use, and computer vision models (such as YOLO) used for traffic analysis. As it covers the iterative, data-driven approach applied to achieve optimal results.

- Chapter 5: Model Evaluation and Testing Plan.
This chapter presents the technical implementation of our project, detailing each step of the system's development, from data preprocessing to model training, testing and evaluation. It also includes the challenges faced and solutions we applied during development.
- Chapter 6: Model Deployment and Integration.
Here in chapter 6, we discuss the results obtained from the project, including model performance metrics and analysis of traffic flow improvements. This chapter evaluates the effectiveness of our Smart traffic management system and compares its performance with existing solutions
- Chapter 7: Conclusion and Future Work.
The final chapter summarizes the whole project, highlights its contributions, and discusses its limitations. Although provides suggestions for future work and potential improvements to the system for wider application and greater impact.

Chapter 2: Literature Review

The field of smart traffic management systems (STMS) has garnered significant attention in recent years, driven by the growing integration of AI into urban infrastructure. Numerous studies and projects have explored various aspects of STMS, leveraging Machine Learning and Computer Vision to optimize traffic flow, enhance road safety, and reduce environmental impact.

This chapter reviews key existing systems, highlights their limitations, and introduces the overall approach for addressing the challenges inherent in traffic management. By examining these aspects, we aim to establish a comprehensive foundation for our project's objectives and methodologies.

2.1 Existing Systems

Through an extensive review of existing literature and applications in the field of smart traffic management systems, several notable implementations have emerged. These systems demonstrate the effectiveness of leveraging advanced technologies, such as artificial intelligence and computer vision, to optimize traffic flow, enhance safety, and reduce environmental impact. Below, we explore some of the most widely recognized and influential examples of these systems:

1. **London's Transportation Intelligent Mobility Management System (TIMMS)**.
   TIMMS aims to future-proof London's transportation network through advanced traffic signal improvements, including transit signal priority, sensors, and video cameras along major corridors. As per the City of London's official announcements, the project is scheduled to run from 2019 to 2027. It incorporates a Transportation Management Centre (TMC), enabling real-time signal timing adjustments and coordination with emergency operations to optimize traffic flow dynamically.

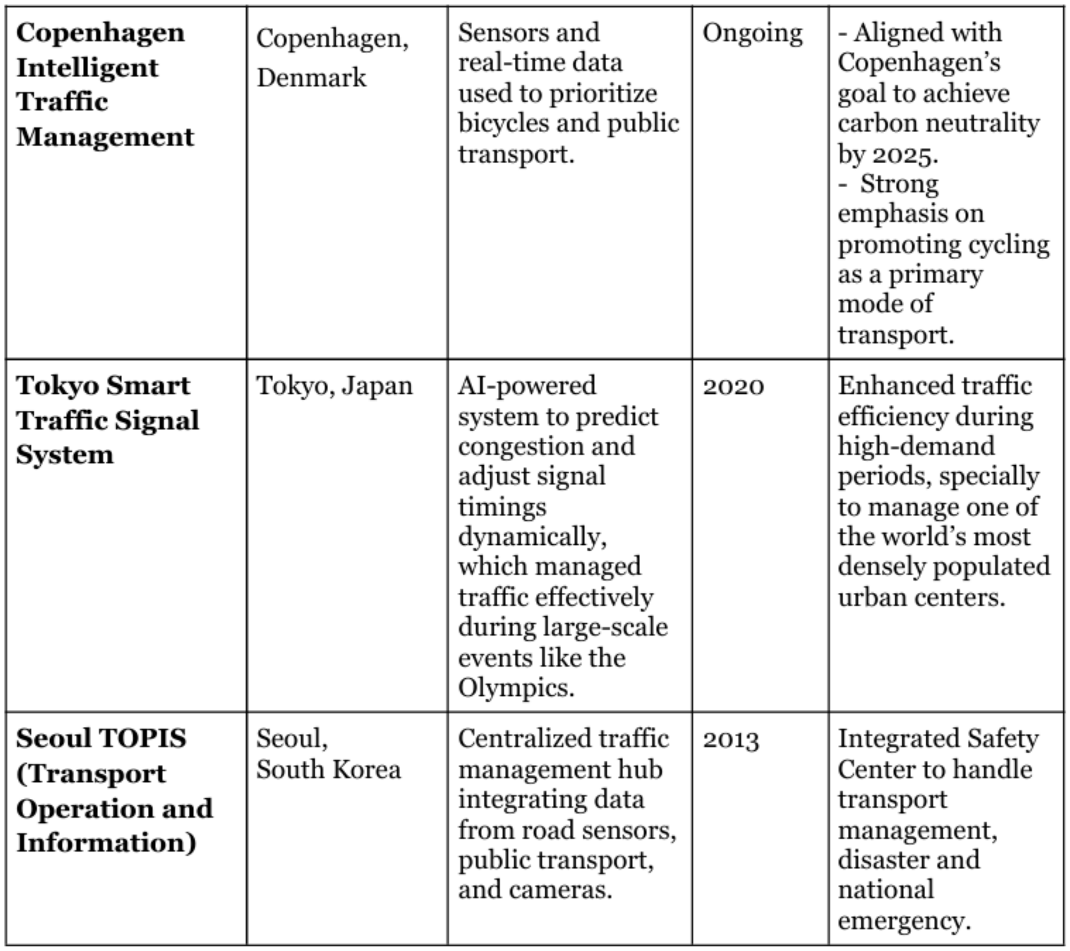
2. **Singapore’s AI-Powered Traffic Management System.**
   Launched in October 2024, Singapore's AI-powered traffic management system utilizes real-time data from traffic cameras and sensors spread across the road network. This system actively monitors vehicle flow, predicts congestion patterns, and adjusts traffic signals dynamically to ensure smooth and efficient traffic management.
3. **The Dubai Intelligent Traffic Systems Centre.**
   Developed by Dubai's Road and Transportation Authority (RTA), this advanced system integrates AI, IoT, and big data to monitor and optimize traffic in real time. It utilizes a network of sensors and cameras to manage traffic flow, enhance safety, and address incidents efficiently. Designed for scalability, the system supports future road network expansions and aligns with Dubai's vision of smart, sustainable urban transportation.
4. **Mumbai's ATSC system.**
   part of the city's broader "Intelli Traffic" initiative, addresses chronic traffic congestion by dynamically adjusting traffic signal timings based on real-time vehicle flow. Using AI, machine learning, and sensor technologies, the system monitors traffic density, predicts patterns, and optimizes signal timings at intersections to ensure smoother flow during peak hours.
   This system has significantly reduced idle times, minimized bottlenecks, and decreased fuel consumption, contributing to lower carbon emissions. By integrating smart solutions into urban infrastructure, the ATSC initiative highlights an efficient and sustainable approach to managing traffic in one of the world's most congested cities.

5. **Amman's Traffic Monitoring Systems.**
   The Greater Amman Municipality (GAM) has implemented centralized monitoring systems that utilize traffic cameras to observe and manage traffic conditions. While effective for basic monitoring, these systems lack advanced AI-driven functionalities, such as adaptive traffic light control, leaving room for technological advancements to address the city's growing traffic challenges.
6. **Los Angeles Automated Traffic Surveillance and Control (ATSAC).**
   ATSAC is one of the most sophisticated traffic management systems in the U.S., managing over 4,500 traffic signals throughout Los Angeles. The system monitors traffic in real time using sensors and cameras, enabling adaptive signal control to minimize congestion. It also integrates public transit prioritization to improve bus and train punctuality.
   ATSAC has significantly reduced travel time, increased intersection efficiency, and contributed to improved air quality in the region.
7. **Barcelona Smart Mobility Project.**
   Barcelona has implemented a comprehensive smart mobility strategy, including a system that monitors traffic using IoT sensors, cameras, and connected devices. The city employs real-time data to manage traffic lights and parking spaces dynamically.
   The system integrates with Barcelona's public transport network and includes bicycle-friendly infrastructure. The system has Reduced traffic congestion and improved urban air quality while promoting sustainable transportation options.
8. **Copenhagen Intelligent Traffic Management.**
   Copenhagen integrates a smart traffic system to prioritize bicycles and public transport. Using sensors and real-time data, the system adjusts traffic signals to reduce delays for cyclists and buses. The system supports Copenhagen's goal to become carbon-neutral by 2025 by encouraging green transportation modes.

9. **Tokyo Smart Traffic Signal System.**
   Tokyo uses an AI-powered traffic signal system that predicts congestion and adjusts light timings in real time. This system supports large-scale events and day-to-day traffic management in one of the world's most densely populated urban centers. The system enhanced traffic flow efficiency and reduced congestion during the Tokyo 2020 Olympics.
10. **Seoul TOPIS (Transport Operation and Information Service).**
   TOPIS acts as a centralized hub for Seoul's traffic management, integrating data from road sensors, public transportation, and traffic cameras. It provides real-time traffic information to drivers and authorities while managing incidents.
   The system uses AI-driven prediction models and big data analytics and enhances mobility, reduces congestion, and improved response to traffic incidents. As Seoul Topis published “In 2013 the Seoul Metropolitan Government established and began to operate the Seoul Integrated Safety Center to handle transport management, disaster and national emergency.”

System	**Location**	**Key features**	**Year of Lunch**	**Unique characteristics**
TIMMS	London, UK	Advanced traffic signal improvements, sensors, video cameras, real-time signal timing, coordination with emergency operations	2019-2027	Features a Transportation Management Centre (TMC) for dynamic traffic optimisation and future scalability.
AI-Powered Traffic Management System	Singapore	Real-time data from cameras and sensors, AI to monitor flow, predict congestion, and adjust signals dynamically	2024	Utilises AI for real-time traffic prediction and adjustments across the network.
Intelligent Traffic Systems Centre	Dubai, UAE	AI, IoT, big data, real-time monitoring, incident management, scalable for future expansions	Ongoing	Integrates smart technologies for seamless scalability and aligns with a sustainable urban vision.
ATSC	Mumbai, India	AI and machine learning-based signal timing, real-time vehicle flow monitoring, pattern prediction, reduced emissions	Ongoing	Significantly reduces idle times and environmental impact, focusing on efficiency and sustainability.

Traffic Monitoring Systems	Amman, Jordan	Centralised traffic monitoring using cameras, manual observation	Ongoing	Basic monitoring capabilities, lacks AI-driven functionalities, highlighting opportunities for upgrades.
Los Angeles Automated Traffic Surveillance and Control (ATSAC)	Los Angeles, USA	- Real-time traffic monitoring via a network of sensors and cameras. - Adaptive signal control to dynamically optimize traffic flow.	August 2016	One of the largest and most sophisticated systems in the U.S., managing over 4,500 signals.
Barcelona Smart Mobility Project	Barcelona, Spain	- Integration of IoT sensors and connected devices for traffic and parking management. - Infrastructure supporting sustainable transportation modes, including bicycles and public transport.	Ongoing	A comprehensive approach combining traffic management with sustainability goals.



Copenhagen Intelligent Traffic Management	Copenhagen, Denmark	Sensors and real-time data used to prioritize bicycles and public transport.	Ongoing	- Aligned with Copenhagen's goal to achieve carbon neutrality by 2025. - Strong emphasis on promoting cycling as a primary mode of transport.
Tokyo Smart Traffic Signal System	Tokyo, Japan	AI-powered system to predict congestion and adjust signal timings dynamically, which managed traffic effectively during large-scale events like the Olympics.	2020	Enhanced traffic efficiency during high-demand periods, specially to manage one of the world's most densely populated urban centers.
Seoul TOPIS (Transport Operation and Information)	Seoul, South Korea	Centralized traffic management hub integrating data from road sensors, public transport, and cameras.	2013	Integrated Safety Center to handle transport management, disaster and national emergency.

While current systems provide improvements over traditional signal control, most of them rely solely on visual inputs or fixed logic without adaptation. In contrast, our proposed system introduces a novel approach by combining visual and audio detection using YOLOv11 and YAMNet. Furthermore, it incorporates a hybrid control logic that begins with fixed safety rules and evolves over time through reinforcement learning. This integration makes our system uniquely capable of adapting to real-time traffic and emergency conditions, filling a critical gap in the current landscape of intelligent traffic systems.

2.2 Limitations of Existing Systems

While smart traffic management systems represent a significant technological advancement, they are not without their challenges. Despite their transformative potential and essential role in modern urban planning, these systems face various limitations that impact their design, deployment, and overall effectiveness. Below, we explore some of the most prominent limitations associated with existing systems.

1. Infrastructure Dependence: Requires advanced sensors and cameras, challenging for underdeveloped cities. Smart traffic management systems rely heavily on sophisticated hardware, such as high-resolution cameras, radar, lidar sensors, and IoT devices, to collect real-time data. Underdeveloped cities or regions with limited infrastructure may lack the resources to install and maintain these technologies. For example, rural areas or economically constrained municipalities may not have the electrical grid, internet connectivity, or technical expertise needed to support such systems. This dependence creates a barrier to adoption, as the initial setup requires significant investment in physical infrastructure, which may be impractical for areas with outdated or minimal road networks.

2. High Costs: Maintenance and operation are expensive. The deployment and ongoing operation of smart traffic systems involve substantial financial costs. Initial expenses include purchasing and installing cameras, sensors, and computing hardware, as well as integrating software platforms. Beyond setup, regular maintenance—such as repairing malfunctioning sensors, updating software, and ensuring cybersecurity—adds to the operational burden. For instance, a single high-quality traffic camera can cost thousands of dollars, and a city-wide system may require hundreds of such devices. Additionally, skilled personnel are needed to monitor and manage the system, further increasing labor costs. These expenses can strain municipal budgets, particularly in cities with limited funding.

3. Data Challenges: Connectivity issues and inaccurate data affect performance. Smart traffic systems depend on continuous, real-time data from multiple sources (e.g., cameras, sensors, GPS). Connectivity issues, such as network outages or weak internet infrastructure, can disrupt data transmission, leading to delays or incomplete information. Additionally, inaccurate data—caused by sensor malfunctions, environmental interference (e.g., fog affecting cameras), or human errors—can result in flawed decision-making. For example, if a camera misidentifies a vehicle or a sensor fails to detect traffic flow, the system may adjust traffic signals incorrectly, worsening congestion. Ensuring data reliability and robust connectivity is a significant challenge, especially in areas with unstable networks.
4. Unexpected Scenarios: AI struggles with disasters or protests. AI-driven traffic systems are typically trained on predictable traffic patterns but may struggle to adapt to extraordinary events, such as natural disasters (e.g., floods, earthquakes), large-scale protests, or major accidents. These scenarios introduce unpredictable variables, like road closures, erratic driver behavior, or blocked camera views, which the AI may not be equipped to handle. For instance, during a protest, the system might fail to reroute traffic effectively if crowds block intersections. Such situations often require human intuition and contextual understanding, necessitating manual intervention to override or guide the AI, which can delay responses and reduce system reliability.
5. Driver Compliance: Resistance to system guidance reduces effectiveness. The success of smart traffic systems depends partly on drivers following system recommendations, such as suggested routes or signal timings. However, some drivers may ignore guidance due to distrust in technology, preference for familiar routes, or lack of awareness.

6. Integration Issues: Compatibility with legacy systems is complex. Many cities still use older traffic management infrastructure, such as manual or fixed-timing traffic lights, which are not designed to interface with modern AI-based systems. Integrating smart traffic systems with these legacy setups is technically challenging and costly. For instance, adapting existing traffic lights to communicate with a centralized AI system may require hardware upgrades or custom software interfaces. Compatibility issues can lead to incomplete system coverage, where only certain intersections are smart enabled, reducing overall effectiveness. Additionally, integration with other municipal systems (e.g., public transit or emergency services) adds further complexity, requiring standardized protocols and extensive testing.

7. Privacy Concerns: Data collection raises security and ethical issues. Smart traffic systems collect vast amounts of data, including video footage, vehicle GPS locations, and traffic patterns, which can raise significant privacy and security concerns. For example, cameras capturing license plates or tracking vehicle movements could inadvertently collect personal information, raising ethical questions about surveillance. If data is not properly anonymized or secured, it could be vulnerable to breaches or misuse, eroding public trust. Compliance with data protection regulations, such as GDPR, adds complexity, as systems must ensure encryption, anonymization, and transparent data usage policies. Balancing effective traffic management with privacy protection is a critical challenge.

8. Environmental Impact: Hardware production has a carbon footprint. While smart traffic systems aim to reduce vehicle emissions by optimizing traffic flow, the production, deployment, and maintenance of their hardware components—such as cameras, sensors, and servers—generate a significant environmental footprint. Manufacturing these devices involves energy-intensive processes and materials like rare earth metals, contributing to carbon emissions and electronic waste. Additionally, the energy required to power these systems, especially in large-scale deployments, can offset some environmental benefits. For example, a city-wide network of sensors and servers running 24/7 consumes considerable electricity, challenging the system's sustainability claims unless renewable energy sources are used.

9. Cultural Barriers: Adoption varies due to driving habits. The effectiveness of smart traffic systems can be influenced by local driving cultures and behaviors. In regions where drivers are less accustomed to following traffic rules or relying on technology, adoption may be slow. For instance, in some areas, drivers may disregard traffic signals or lane markings, disrupting the system's ability to manage flow efficiently. Cultural attitudes toward technology also play a role; communities with low trust in automated systems may resist smart traffic solutions.

2.3 Overall solution Approach

1. Using durable, low-maintenance devices with predictive maintenance. STMS uses robust, IP-rated devices to reduce repairs. Predictive maintenance algorithms flag potential failures, minimizing downtime and costs, making it suitable for resource-limited cities.
2. Leveraging edge computing for data processing during connectivity issues. Edge computing enables local data processing on devices, ensuring operation during network outages. This reduces latency and improves reliability in areas with unstable connectivity.
3. Incorporating fail-safe protocols for unexpected scenarios. Fail-safe protocols allow fallback modes or human intervention during disasters. Diverse training data enhances AI adaptability, ensuring safety in unpredictable situations.
4. Designing modular APIs for integration with existing systems. Modular APIs enable compatibility with legacy traffic systems, allowing gradual upgrades. Extensive testing ensures seamless data exchange, reducing integration complexity.
5. Implementing encryption and anonymization for privacy. STMS uses encryption and anonymizes data (e.g., blurring license plates) to protect privacy. GDPR compliance and audits prevent breaches, building public trust.
6. Using energy-efficient hardware to reduce environmental impact. Energy-efficient sensors and solar-powered devices lower power use. Optimized algorithms reduce computational demands, aligning with sustainability goals.
7. Providing user-friendly interfaces and public engagement for acceptance. Intuitive dashboards and public campaigns promote system use. Engagement educates communities, addressing distrust and aligning local driving habits for better compliance.

Chapter 3: Requirement Engineering and Analysis.

Requirement Engineering Overview

Requirement engineering is a systematic process to define, document, and maintain the requirements for the Smart Traffic Management System **(STMS)**. This process ensures the system aligns with stakeholder expectations, operates effectively in dynamic urban environments, and delivers real-time, intelligent traffic management. By integrating computer vision (YOLOv11) and audio processing (YAMNet), the STMS addresses traffic congestion, emergency vehicle prioritization, and adaptive signal control, meeting both functional and non-functional needs.

Stakeholder Identification

1. Stakeholders are individuals or entities with a vested interest in STMS's performance. Key stakeholders include:
2. Traffic Departments: Responsible for traffic regulation and management. They require real-time monitoring, congestion analysis, and tools to prioritize emergency vehicles (e.g., ambulances) and optimize signal timing.
3. City Planners: Use traffic data for urban planning, infrastructure expansion, and smart city initiatives. They need insights into traffic patterns and density for long-term decision-making.
4. Road Users: Drivers, pedestrians, and cyclists expect reduced congestion, improved safety, and timely traffic updates, especially during emergencies.
5. Public Transport Authorities: Rely on real-time traffic data to optimize bus and transit schedules, integrating with adaptive signal control for smoother operations.
6. Government and Policymakers: Leverage traffic data to enforce regulations, reduce emissions, and advance smart city goals, ensuring compliance with privacy and environmental standards.
7. Emergency Services: Require prioritized routing for ambulances, supported by real-time vehicle and siren detection to minimize response times.

Functional Requirements

Functional requirements specify the core capabilities of the STMS to meet stakeholder needs:

1. Real-time Traffic Monitoring:
    a. Collects data from CCTV cameras, drones, and IoT sensors for live vehicle detection and classification (cars, buses, trucks, motorcycles, ambulances).
    b. Detects traffic light states (red, green) using YOLOv11 and OpenCV for color classification.
    c. Provides live updates on vehicle density, traffic flow, and incidents within a Region of Interest (ROI).
2. Emergency Vehicle Detection and Prioritization:
    a. Identify ambulances using YOLOv11 trained on the Ambulance Detection Dataset from RoboFlow.
    b. Detects emergency sirens via a YAMNet-based model processing 16 kHz audio, enabling immediate traffic signal adjustments (e.g., green light for ambulances).
    c. Activates emergency protocols to prioritize ambulance passage, overriding standard signal timing.
3. Adaptive Traffic Signal Control:
    a. Dynamically adjusts traffic light durations based on vehicle density, time of day (day/night), and emergency vehicle presence, using the decision-making system.
    b. Supports actions like extending green lights for heavy traffic (up to 90 seconds) or shortening red lights for light traffic.
4. Congestion Analysis and Decision Support:
    a. Calculates vehicle density within the ROI using the calculate_density function to assess congestion levels.
    b. Integrates external data (e.g., weather, GPS, accident reports) to predict congestion and suggest proactive measures like rerouting.

5. Contextual Advertising Control:
    a. Manages roadside ad displays based on traffic conditions, pausing ads during heavy traffic or emergencies and resuming during light traffic.
6. Data Storage and Analytics:
    a. Stores processed video frames, audio data, and metadata in MongoDB for future retraining and traffic pattern analysis.
    b. Generates human-readable outputs (e.g., vehicle counts, signal commands, reasons) for stakeholder dashboards.

Non-Functional Requirements

Non-functional requirements ensure the STMS's operational quality and reliability:

1. High Reliability and Scalability:
    a. Operates 24/7 with minimal downtime, handling high volumes of video and audio data.
    b. Scales to process multiple camera feeds and intersections using modular architecture and Flask-based deployment.
2. Low Latency for Real-time Processing:
    a. Achieves inference times of ~1.7 ms per image and total latency of ~2.9 ms per frame, supporting >30 FPS for real-time applications.
    b. Ensures rapid response for time-critical tasks like emergency vehicle prioritization.
3. Data Security and Privacy:
    a. Encrypts sensitive data (e.g., GPS logs, license plates) and anonymizes personal identifiers to comply with GDPR and other regulations.
    b. Uses secure APIs and communication protocols to protect against cyber threats.

4. Usability and Maintainability:
    a. Provides a user-friendly web interface (HTML, CSS, JavaScript) for stakeholders to upload videos, view analytics, and receive alerts.
    b. Modular design allows independent updates to YOLOv11, YAMNet, or decision logic without system-wide disruption.
5. Energy Efficiency:
    a. Optimize computational resources (e.g., GPU acceleration on NVIDIA RTX 3080) to minimize power consumption while maintaining performance.
    b. Supports potential edge deployment with model compression for lower energy use.

3.1 Data Understanding and Preparation

- Data Characteristics:
- Traffic data for the STMS is characterized by the 4Vs:
- Volume: High data volumes from continuous CCTV feeds (gigabytes daily per camera), IoT sensors, and audio streams (terabytes city-wide).
- Velocity: Real-time data streams, with vehicle detections and siren classifications updated every few seconds.
- Variety: Includes structured data (sensor logs, vehicle counts), unstructured data (video, audio), and semi-structured data (JSON from APIs).
- Veracity: Data accuracy is critical, as sensor errors, misaligned timestamps, or noisy audio can lead to incorrect decisions.

3.1.1 Data Collection Methods and Sources

Effective data collection is foundational to the success of a Smart Traffic Management System (STMS). It involves gathering diverse traffic-related data from various sources, both in real-time and historically, to provide a comprehensive view of traffic conditions. The data collected serves as the basis for all predictive, analytical, and operational processes within the system.

Key Data Sources:Data collection is foundational to the STMS, integrating diverse sources for comprehensive traffic insights:

- Primary Data Sources:
    - **Vehicle Detection Dataset**: Capture real-time video for vehicle and traffic light detection (e.g., UA-DETRAC Dataset, Traffic Lights Dataset).
    - **Ambulance Detection Dataset**: Labeled images from RoboFlow for training YOLOv11 to identify ambulances.
    - **Ambulance Siren Detection Dataset**: Audio clips (.wav) labeled as “siren” (1) or “others” (0) for YAMNet training.

Data Quality Considerations:

- **Accuracy**: Ensuring that sensors, cameras, and external data sources provide high-quality, accurate data for effective analysis.
- **Completeness**: Data gaps can lead to incorrect analysis, so ensuring coverage across all regions and timeframes is critical.
- **Consistency**: Different data sources need to be synchronized to avoid discrepancies in timestamps or geographic locations.

3.1.2 Data Preprocessing Techniques and Cleaning

Preprocessing ensures data is clean, structured, and model-ready. Techniques are tailored to the STMS's video and audio inputs:

- Preprocessing Techniques:
  - Normalization: Scales image pixel values and audio amplitudes (e.g., Min-Max scaling for images, z-score for audio features).
  - Encoding: Converts vehicle types (e.g., "Car", "Ambulance") to numerical labels for YOLOv11 and YAMNet.
  - Resampling: Standardizes audio to 16 kHz using librosa and video frames to 640×640 or 896×896 for YOLOv11.
  - Data Augmentation:
    - Images: Applies rotation, flipping, brightness changes, and blurring UA-DETRAC and Traffic Lights datasets.
    - Audio: Adds noise or pitch shifts to siren clips for robustness.
  - Annotation Review: Converts labels to YOLO format (bounding boxes, class IDs) and validates siren labels ("siren" = 1, "others" = 0).
  - Temporal Dimension Reduction: Averages YAMNet embedding over time to create fixed-size audio feature vectors.
  - ROI Filtering: Uses is_inside_roi() to focus on relevant road segments.
- Cleaning Techniques:
  - Missing Data: Removes corrupt images or audio clips; imputes minor gaps using interpolation for sensor data.
  - Outlier Detection: Applies Z-scores to filter anomalous vehicle counts or audio features; uses Isolation Forests for complex outliers.
  - Duplicate Elimination: Removes redundant frames or audio clips using hash-based checks.

    - Noise Reduction:
        - Images: Applies histogram equalization and sharpening to correct motion blur.
        - Audio: Uses Gaussian filters to stabilize siren waveforms.
    - Standardization: Adopts ISO 8601 for timestamps and kilometers per hour for speed.
    - Label Validation: Manually reviews a subset of annotations and uses scripts to detect mislabeled data.
- Outcome:
    - Consistent, high-quality data for training and inference.
    - Reduced noise and redundancy for efficient processing.
    - Enhanced model robustness through augmentation and cleaning.

3.1.3 Data Integration and Transformation

Data integration and transformation consolidate multi-modal data into a unified, model-ready format:

- Data Integration:

  Sources:

  - Video feeds.
  - Audio clips (siren detection).

  Tasks:

  - Centralized Repository: Stores data in MongoDB for structured and unstructured formats.
  - Real-time Streaming: Uses Apache Kafka to integrate live feeds.
  - Data Alignment: Synchronizes data by timestamps and geo-coordinates.

  Tools:

  - Apache Kafka for streaming.
  - MongoDB for storage.
  - Apache Airflow for workflow orchestration.

- Data Transformation:
    - Cleaning and Preprocessing:
        - Handles missing values and standardizes formats (e.g., JPEG for images, 16 kHz for audio).
        - Extracts frames from MP4 videos using OpenCV.
    - Enrichment:
        - Adds weather and event data to contextualize traffic patterns.
        - Computers are vehicle density and traffic light states.
    - Feature Extraction:
        - Generates bounding boxes and class IDs from YOLOv11.
        - Extracts YAMNet embeddings for siren detection.
        - Calculates temporal features (e.g., peak hours) and spatial features (e.g., ROI density).
    - Tools:
        - OpenCV for video preprocessing.
        - Librosa for audio processing.
        - Pandas/NumPy for data manipulation.
        - Scikit-learn for feature scaling.
- Challenges and Solutions:
    - Challenge: Real-time data consistency.
        - Solution: Uses Kafka for synchronized streaming.
    - Challenge: Scalability.
        - Solution: Leverages cloud storage (e.g., AWS S3) and modular pipelines.

3.2 Feature Engineering and Selection

Feature engineering and selection create and refine features to optimize STMS model performance:

- Feature Engineering:
- Video Data:
    - **Features**: Vehicle counts, types, density, speed, bounding box coordinates.
    - **Techniques**: YOLOv11 for object detection, optical flow for speed estimation.
- Audio Data:
    - **Features**: Siren presence, audio embeddings.
    - **Techniques**: YAMNet for feature extraction, temporal averaging for fixed-size vectors.
- Examples:
    - Temporal: Hour, day, peak/off-peak indicators.
    - Spatial: ROI density, congestion hotspots.
    - Contextual: Weather impact, event-based traffic surges.
- Feature Selection:
- Filter Methods:
    - Correlation analysis to remove redundant features (e.g., vehicle count vs. density).
    - Chi-Square tests for categorical features (e.g., weather vs. congestion).
- Wrapper Methods:
    - Recursive Feature Elimination (RFE) to prioritize impactful features.

- Embedded Methods:
    - LASSO for feature coefficient reduction.
    - Random Forest for feature importance ranking.

- Selected Features:
    - Vehicle density, ambulance presence, siren detection scores, traffic light states, time-of-day, weather conditions.
- Tools:
    - Pandas/NumPy for feature extraction.
    - Scikit-learn for selection.
    - Matplotlib/Seaborn for visualization.
- Outcome:
    - Optimized feature set for accurate congestion prediction and emergency prioritization.
    - Reduced computational load for real-time inference.

3.3 Ethical Considerations and Data Privacy

Ethical Considerations:

- **Equity and Accessibility**: Ensures STMS benefits all communities, avoiding bias in signal prioritization (e.g., favoring affluent areas). Deploys cameras and sensors equitably across urban and underserved regions.
- **Transparency**: Publishes clear documentation on data usage, model decisions, and signal control logic to build public trust.
- **Accountability**: Implements regular audits and monitoring to address errors (e.g., missed ambulance detections) and prevent safety risks.
- **Environmental Impact**: Optimizes traffic flow to reduce emissions while minimizing energy use through efficient GPU inference and potential edge deployment.
- **Public Trust**: Engages communities via public forums to address surveillance concerns and promote acceptance.

Data Privacy Considerations:

- **Data Collection**: Limits data to essential inputs (e.g., vehicle types, siren audio) to minimize privacy risks.
- **Data Anonymization**: Removes identifiable data (e.g., license plates, GPS traces) using masking or hashing techniques.
- **Data Security**: Employs encryption (e.g., TLS for APIs) and secure storage (MongoDB with access controls) to protect against breaches.
- **Consent and User Rights**: Obtains consent for crowdsourced data (e.g., Waze) and allows users to opt out of data collection.
- **Data Retention**: Deletes raw video and audio after processing, retaining only anonymized metadata for analytics.
- **Compliance**: Adheres to GDPR and local privacy laws, ensuring legal and ethical data handling.

- **Balancing Innovation and Privacy**

  STMS leverages AI to enhance urban mobility while prioritizing privacy through anonymization, minimal data collection, and secure processing. Ethical measures ensure fairness, transparency, and accountability, fostering public trust and sustainable deployment.

Chapter 4: Model Development and Architectural Design

4.1 System Architecture Overview

Our aim in this section is to define the high-level structure of the Smart Traffic Management System (STMS) and its core components. Our system is designed to leverage real-time computer vision techniques for efficient traffic monitoring, congestion detection, and adaptive traffic control.

1. High-Level Structure & Key Components

- 1.1 Data Sources (Input)
    - **UA-DETRAC Dataset:** A large-scale dataset containing traffic images. Our aim is to Label the data to include Vehicle types (cars, buses, trucks, motorcycles)
    - **Traffic Lights Dataset:** Used for detecting and classifying traffic signals.
    - **Ambulance Detection Dataset:** Specifically trained to identify emergency vehicles.
    - **Ambulance Siren Detection Dataset:** Specifically trained to differentiate between emergency siren and traffic noise.

Future Deployment (Live Traffic Feeds):

- CCTV Cameras & Drone Footage: Real-time feeds for live traffic analysis.
- External Data Feeds: Weather reports, GPS data, and accident reports.

1.2 Preprocessing Pipeline:

To enhance the quality and usability of input data, the system employs a data preprocessing pipeline that includes:
For Image detection datasets (UA-DETRAC, Traffic Lights, Ambulance Detection):

- Data Cleaning:
    - Removing duplicates and corrupt images.
    - Standardizing image sizes and formats (JPEG).
    - Applying noise reduction, histogram equalization (contrast improvement), and sharpening (to correct motion blur).
- Data Augmentation: Enhancing dataset diversity through transformations like rotation, flipping, brightness changes, and blurring.
- Annotation Review: Converting labeled data into YOLO format for model training.

For Audio Detection Dataset (Ambulance Siren Detection):

- Audio Loading and Resampling
  Load each audio file (.wav) and resample it to 16 kHz. Standardizing the sampling rate is important so all audio inputs are consistent for the model. Using 16,000 samples per second forces all audio to be in the same sample rate, which is a common standard in speech/audio models.
- Organized Directory Structure
  automatic labeling during preprocessing, "siren" (label 1), "others" (label 0)
- Audio Preprocessing for Feature Extraction
  The loaded audio clip is fed into the YAMNet model from TensorFlow Hub. YAMNet is a pre-trained model that maps audio waveforms to embeddings based on the AudioSet ontology.

- Temporal Dimension Reduction
  The embeddings returned are time-dependent (multiple vectors over time), and to make it simpler, we take the mean of the embeddings across the time axis (axis=0). This converts the variable-length audio data into a fixed-size feature vector, making it easier to use in ML models.
- Label Assignment
  "siren" (label 1), "others" (label 0), these labels are stored in the labels list for supervised learning.

1.3 YOLOv11 Model – Traffic Object Detection, Neural Network Classifier - Audio Recognition

The system utilizes YOLOv11, an advanced object detection model, to identify and classify various traffic-related objects:

- Vehicles and Emergency Vehicles: Ambulance detection is enhanced using labeled data from RoboFlow and detection of cars, trucks, buses, and motorcycles.
- Traffic Lights: YOLOv11 detects traffic light objects, while OpenCV libraries are used to classify their colors (red, green).

Also, the system utilizes a neural network to differentiate between sirens and traffic noise using high-dimensional audio features.

1.4 Decision-Making System

The decision system is designed to intelligently manage traffic signals by analyzing real-time vehicle density, detecting emergency vehicles, and adapting to the time of day (day or night). It begins by calculating the density of vehicles within a specific region of interest (ROI) using the calculate_density function, which determines the ratio of vehicles inside the ROI to the total detected vehicles. Depending on the result, the system compares the density to predefined thresholds to determine if traffic is considered light or heavy. Based on this, along with current light status, siren or ambulance presence, and whether it's day or night, the decision_logic function decides how long traffic lights should stay green or red. For example, during heavy traffic at night, the system may extend the green light duration to a maximum of 90 seconds, while light traffic may trigger a shorter green phase. If an ambulance is detected, emergency rules are activated to give it immediate priority by turning the light green. The system also includes safeguards—such as holding all actions if an invalid time mode is provided—to ensure safety and reliability in unexpected scenarios.

Action Mapping: Each decision maps to a specific traffic light duration:

- **Action 1**: Emergency Protocol – Maximum green duration
- **Action 2**: Heavy Traffic – Maximum green duration based on density
- **Action 3**: Normal Flow – Adjust red light based on density
- **Action 4**: Light Traffic – Maximum red duration based on density

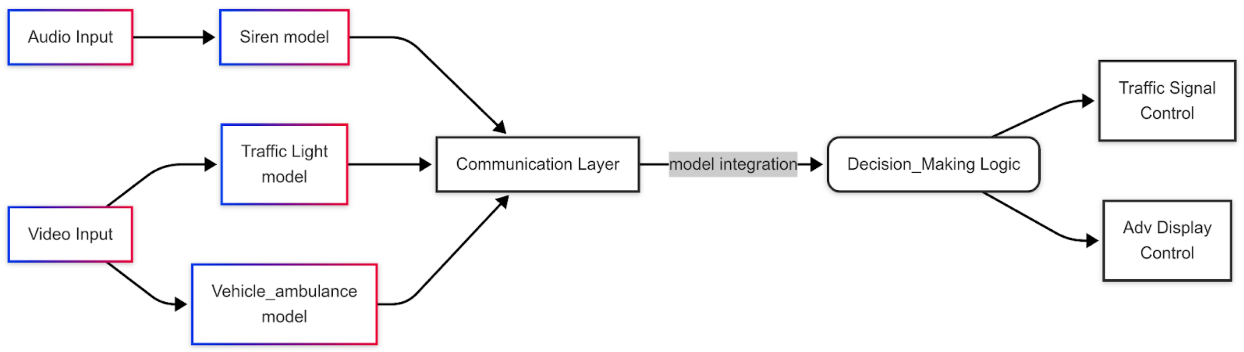
1. Output Layer – Control Actions & Visualization

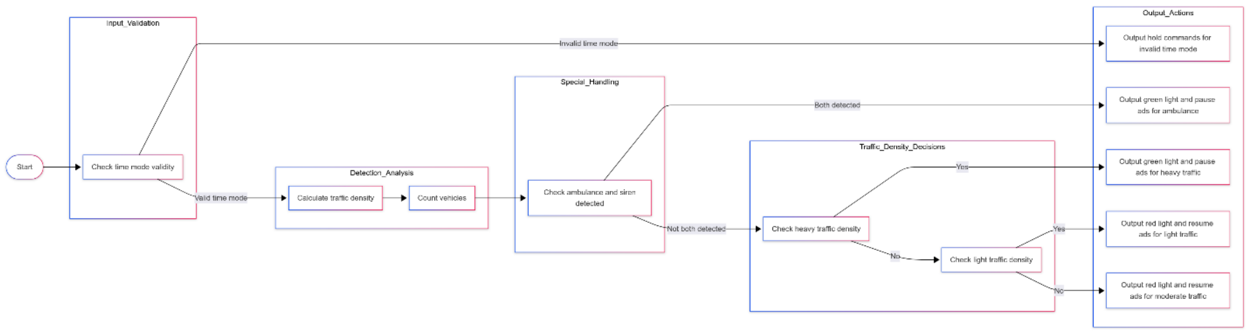
The output layer of this decision-making system is a dictionary that provides clear commands and explanations for traffic control actions. It includes three main fields: traffic_signal_command, which determines whether the traffic light should be green, red, or held based on the current conditions; ad_display_command, which decides whether roadside ads should be shown or hidden depending on traffic flow; and reason, which gives a human-readable explanation of why these actions were chosen.

This structured output enables real-time, intelligent control of traffic systems by considering factors such as vehicle density within a defined region of interest (ROI), emergency vehicle detection, and whether it is day or night. The output ensures that both traffic and public display systems respond dynamically and appropriately to real-world traffic scenarios.

2. Architecture Diagram Overview

The following sequence represents the data flow within the STMS:




Start
Input_Validation
Check time mode validity
Invalid time mode
Valid time mode
Detection_Analysis
Calculate traffic density
Count vehicles
Special_Handling
Check ambulance and siren detected
Both detected
Not both detected
Traffic_Density_Decisions
Check heavy traffic density
Yes
No
Check light traffic density
Yes
No
Output_Actions
Output hold commands for invalid time mode
Output green light and pause ads for ambulance
Output green light and pause ads for heavy traffic
Output red light and resume ads for light traffic
Output red light and resume ads for moderate traffic

4.2 Data Pipeline Design

Our pipeline is designed to process video files for intelligent traffic monitoring by detecting vehicles, traffic lights, and emergency sirens. The system uses a combination of computer vision and audio analysis to extract contextual insights from urban traffic scenes. Here's a breakdown of the data flow and processing stages:

1. Model Initialization: Three pretrained models are loaded at the start of the pipeline:

- Vehicle Detection Model (YOLOv11): Detects and classify vehicles (Car, Motorcycle, Bus, Truck, Ambulance) using a custom-trained YOLO model.
- Traffic Light Detection Model (YOLOv11): Detects traffic light states (e.g., red, green).
- Siren Detection Model (TensorFlow): A custom YAMNet model that classifies siren sounds in audio clips.

2. Video and Audio Input

- Input Format: MP4 video file.
- The video is read frame-by-frame for object detection.
- Audio Extraction:
    - The audio track is extracted from the MP4 file using moviepy.
    - The extracted audio is saved temporarily as a .wav file for analysis.

3. Audio Processing (Siren Detection)

- The .wav audio is loaded using librosa and sampled to 16kHz.
- The waveform is passed to the siren model to check for emergency vehicle presence.
- If any siren-related class has a confidence score > 0.1, the presence of a siren is confirmed.

4. Frame-by-Frame Video Processing

- Each video frame is passed through:
    - The vehicle detection model to detect and classify vehicles.
    - The traffic light model to recognize light status.
- Bounding boxes and class IDs are extracted for objects detected.

5. Region of Interest (ROI) Filtering

- The predefined ROI is used to focus analysis on the main traffic lane.
- For each detection, the pipeline checks if the object falls within the ROI using geometric conditions.

6. Feature Extraction (Counts & Categories)

- Vehicles are counted by category (car, motorcycle, bus, truck).
- These counts are used for further analysis like congestion estimation and decision making.

This scalable design ensures the system can be extended easily with new models or sensors in the future. It also allows independent testing and optimization of video and audio branches of the pipeline.

4.3 Model Architecture and Design

This system integrates three specialized AI models, each targeting a specific modality (vision or audio), forming a multi-modal architecture for traffic monitoring and emergency detection. The overall model design is modular and follows a multi-input, multi-task approach:

1. Vehicle and Ambulance Detection Model

Type: YOLOv11
Architecture Details: Custom-trained on a dataset of 5 vehicle classes: Car, Motorcycle, Bus, Truck, and Ambulance.
Outputs: Bounding boxes (x, y, width, height) and Class ID
Use in System:

- Runs on every video frame
- Used to determine vehicle types and count
- Coordinates are passed to a Region of Interest (ROI) filter for density calculations

2. Traffic Light Detection Model

Type: YOLOv11
Architecture Details: Similar structure to the vehicle model but trained specifically for detecting traffic lights, detects light state based on object detection of red or green lights.

Use in System:

- Helps determine the current state of the traffic signal in the scene
- Used in the decision-making logic to decide whether to override traffic signals (e.g., in emergency cases)

3. Siren Detection Model

Type: Custom YAMNet-based Model (TensorFlow)
Architecture Details:

- YAMNet is a convolutional neural network (CNN) pre-trained on AudioSet.
- The model is fine-tuned to detect siren sounds using a 1D waveform input.
- Input: 16kHz mono waveform of up to 10 seconds
- Output: Array of class probabilities (specifically 5 siren-related classes)

Use in System:

- Determines the presence of emergency sirens
- If detected, prioritizes emergency vehicle passage in the logic

Model Design Philosophy

- Modularity: Each model is independent and replaceable, allowing flexible upgrades.
- Real-time Capability: Models chosen are optimized for speed to support live video processing.
- Multi-Modal Sensing: Combines vision (YOLO) and audio (YAMNet) for robust detection.
- ROI-based Filtering: Spatial filtering ensures only relevant detections are considered (e.g., inside the road area).

4.4 Training Process

The training process for three core components of the AI-based Smart Traffic Monitoring System: Traffic Light Detection, Vehicle & Ambulance Detection, and Ambulance Siren Detection. Each model was trained independently using different frameworks and tools tailored to the data type (visual or audio).

Traffic Light Detection Model (YOLOv11):

The model used is YOLO11, the extra-large version from the Ultralytics YOLO family, known for its high performance and accuracy in object detection tasks. The training process was executed on a high-performance machine with GPU acceleration enabled.

- **Framework**: Ultralytics YOLO (PyTorch-based)
- **Dataset**: Defined by traffic.yaml with annotated images for red, yellow, and green traffic lights.
- **Input size**: Images resized to 896×896 for high-resolution detection.
- Epochs: 100
- **Batch size**: 32 (balanced with GPU memory usage)
- **Device**: CUDA-enabled GPU
- **Cache**: Enabled to load images into RAM, improving training speed.
- **Augmentations**: Enabled to boost generalization.
- **Mixed Precision (AMP)**: Enabled for improved speed and memory efficiency.
- **Early Stopping**: Added via patience=20 to halt training if no improvement is observed.
- **Seed**: Set to ensure reproducibility.
- **Saving**: Save model only at the end

Vehicle and Ambulance Detection Model (YOLOv11):

This model also uses YOLOv11, optimized for detecting multiple vehicle classes, including emergency vehicles.

- Epochs:100
- **Batch size**: 32 (faster training with sufficient VRAM)
- **Input size**: 640×640 for balanced speed and accuracy.
- **Augmentation**: Enabled for better robustness in different traffic scenes.
- **Early Stopping**: Configured with patience=10.
- **Mixed Precision (AMP)**: Enabled for improved speed and memory efficiency.
- **Cache and Seed**: Used for speed and reproducibility.
- **Seed**: Set to ensure reproducibility.
- **Saving**: Save model only at the end

Ambulance Siren Detection Model (Keras, Audio Classification):

This model uses a fully connected neural network architecture built in TensorFlow/Keras to classify siren sounds from extracted features.

- **Framework**: TensorFlow/Keras
- **Input**: Pre-extracted audio features (features.npy) from siren and non-siren clips.
- **Dataset split**: 80% training, 20% testing
- **Imbalanced data**: Addressed with class_weights computed using sklearn.
- Architecture:
    - Dense layers with dropout for regularization.
    - Final layer uses softmax or sigmoid (based on binary/multiclass classification).
- **Training strategy**: Early stopping and class weighting applied for better generalization.

4.5 Hyperparameter Tuning

Each model was tuned to balance performance, generalization, and training efficiency. Below are the key hyperparameters and the rationale behind their values:

YOLOv11-based Models (Traffic Light, Vehicle, Ambulance):

Hyperparameter	**Value**	**Reason**
lr0	0.0015	A moderate learning rate for stable convergence.
lrf	0.1	Lower final LR helps the model settle smoothly
weight_decay	0. 0005	Prevents overfitting by penalizing large weights.
momentum	0. 937	Keeps gradients moving in the right direction.
optimizer	AdamW	Combines benefits of Adam with better regularization
batch	32	Adjusted based on available VRAM (24GB+).

imgsz	640	Chosen based on model type and desired accuracy.
augment	True	Improves generalization and robustness.
patience	20	Stop training early if validation doesn’t improve.
seed	42	Ensures reproducibility across runs.

Audio Classification Model:

Hyperparameter	Current Value / Usage	Description
Siren Class Indices	[317, 318, 319, 390, 391]	Audio is resampled to 16kHz before inference.
Score Threshold	> 0.1 used during inference decision	specific siren-related classes from Audio Set.
Clip Length (Duration)	10 seconds-waveform = 16000 * 10	It is used to determine if a siren is detected from class probabilities.

By adjusting these hyperparameters through multiple experiments and observing validation performance, the models were tuned for real-time accuracy, speed, and robustness—crucial for smart traffic systems in real-world deployment.

4.6 Validation Strategy

The validation strategy is a crucial part of model training, as it helps monitor how well the model is performing on unseen data during the training process. Here’s how validation is handled (or not) in your current code setups, and what can be improved or recommended:

YOLOv11 Models (Traffic Light & Vehicle/Ambulance Detection)

The model training leverages a hold-out validation strategy defined by a dataset configuration YAML file (vehicle-ambulance.yaml). This file specifies the paths to both the training and validation datasets. The validation process takes place during training at the end of each epoch to assess the model's generalization performance.

How It Works:

- Dataset Split: The file explicitly separates training and validation data.
- Evaluation Timing: The model is evaluated on the validation set after every epoch using performance metrics such as mAP50, precision, recall, and loss.
- Early Stopping: Configured via patience=20, the training process will stop early if validation metrics do not improve for 20 consecutive epochs—preventing overfitting and saving time.
- No Cross-Validation: This setup does not use k-fold cross-validation. It is a single split (train vs. val), typical for deep learning pipelines due to computational costs.

Benefits of This Strategy

- Simple and efficient.
- Validates generalization without complex resampling.
- Combines well with early stopping to avoid unnecessary overfitting.

Ambulance Siren Detection (TensorFlow)

Also, in Ambulance Siren Detection (TensorFlow) code embeds a validation strategy internally during training. By using manual validation split from training data and Early Stopping to monitor performance and prevent overfitting.

Chapter 5: Model Evaluation and Testing Plan

5.1 Testing Plan Overview

This chapter outlines the comprehensive strategy adopted for evaluating the performance and functionality of the proposed YOLOv8-based object detection system for emergency vehicle recognition. The testing plan is designed to ensure that the model meets the predefined objectives in terms of accuracy, speed, and reliability. The evaluation process includes both model-level and system-level validation, using established metrics and benchmarks to assess detection accuracy, generalization ability, and operational robustness in real-world scenarios.

5.1.1 Unit Testing

Each component in the system pipeline was tested in isolation to ensure functional correctness:

- YOLOv11 vehicle & ambulance model: Verified bounding box accuracy, class label correctness.
- YOLOv11 traffic light model: Checked detection of traffic lights and accurate classification of light color.
- Siren detection model: Tested on clean and noisy audio clips to verify audio classification accuracy.

5.1.2 Integration Testing

Inter-module interactions were tested to ensure seamless data flow:

- Verified the unified output structure from all models and their compatibility with the Decision Engine.
- Ensured temporal alignment in the decision logic (e.g., detecting both ambulance and green light before decision to clear path).

5.1.3 Performance Testing

Simulated different traffic scenarios and workloads:

- Processed video files of varying lengths and resolutions.
- Measured inference speed under single-threaded and concurrent processing.
- Introduced real-time video simulation to evaluate systems under time-sensitive conditions.

5.2 Evaluation Metrics

5.2.1 Model Performance Metrics

Vehicle & Ambulance model:

Metric	**Value**	**Interpretation**
mAP@0.5	0.945	Excellent. This shows that on average, the model detects 94.5% of the time at an IoU threshold of 0.5.
mAP@0.5:0.95	0.871	Very strong. This is a more challenging metric and shows that our model performs well across stricter IoU thresholds.
Precision	0.934	High precision indicates low false positives.
Recall	0.948	High recall means the model is effectively identifying most instances (low false negatives).
Inference Speed	1.7ms/images	Very fast – ideal for real-time application
Classes with highest mAP	Ambulance (0.976)	Our model detects ambulances with very high accuracy – which is perfect since it's a core focus.

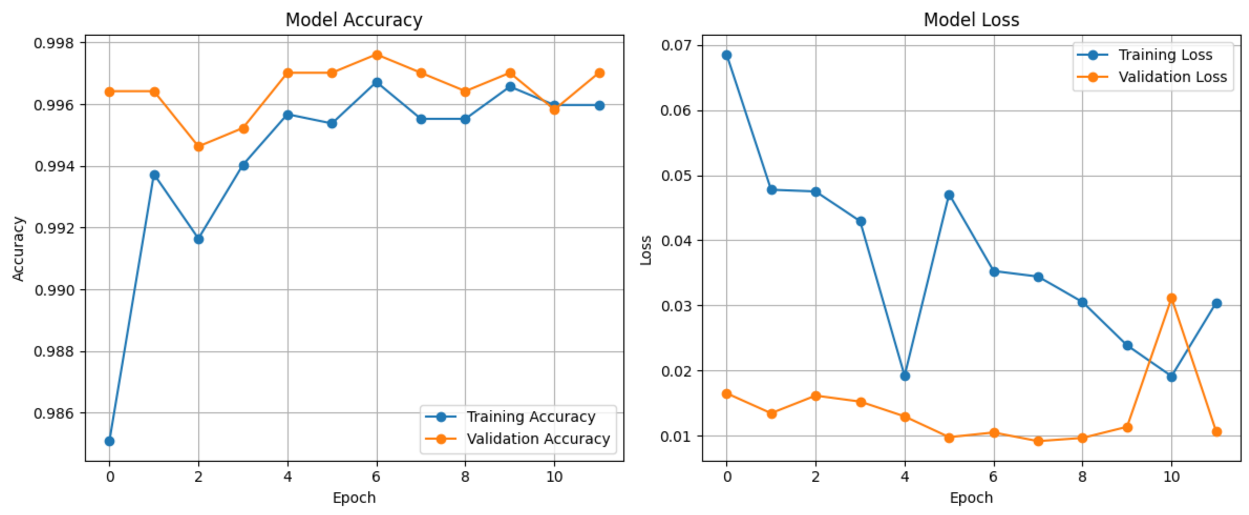
Traffic Light Detection Model:

Metric	**Value**	**Interpretation**
Precision	95.4%	low false positives — model avoids guessing wrong
Recall	92.5%	low false negatives — it rarely misses a detection
mAP@0.5	97.4%	High accuracy when IoU > 0.5
mAP@0.5:0.95	87.5%	Still strong across stricter thresholds (harder to achieve)

Siren Detection Model:

As it shows in the following charts, the model is not only learning well but also generalizing across validation data.

- Training accuracy: ~99.5–99.7%
- Validation accuracy: ~99.6–99.7%
- Validation loss is very low (~0.01) and stays that way after epoch 4.



5.3 Evaluation Methodology

The evaluation methodology is composed of multiple stages:

- Dataset Split:
  The dataset is divided into training and validation sets using a hold-out validation strategy as defined in the vehicle-ambulance.yaml file. This ensures consistent separation between model training and testing data.

- Validation Strategy:
  Evaluation is conducted at the end of every epoch on the validation dataset to monitor model performance and guide early stopping (set to a patience of 20 epochs).

- Testing Environment:
  All experiments are conducted on a GPU-enabled machine using PyTorch via the Ultralytics YOLOv8 framework to ensure fast and reproducible results.

- System Testing:
  Beyond model evaluation, the overall system is tested in real-time scenarios to assess frame rate, object tracking stability, and detection consistency under varying lighting and environmental conditions.

5.4 Model Performance Evaluation

5.4.1 Performance on Training Data

During training, the YOLO11 model demonstrated strong convergence behavior, with steadily decreasing loss values over the epochs. The following metrics were recorded on the training dataset:

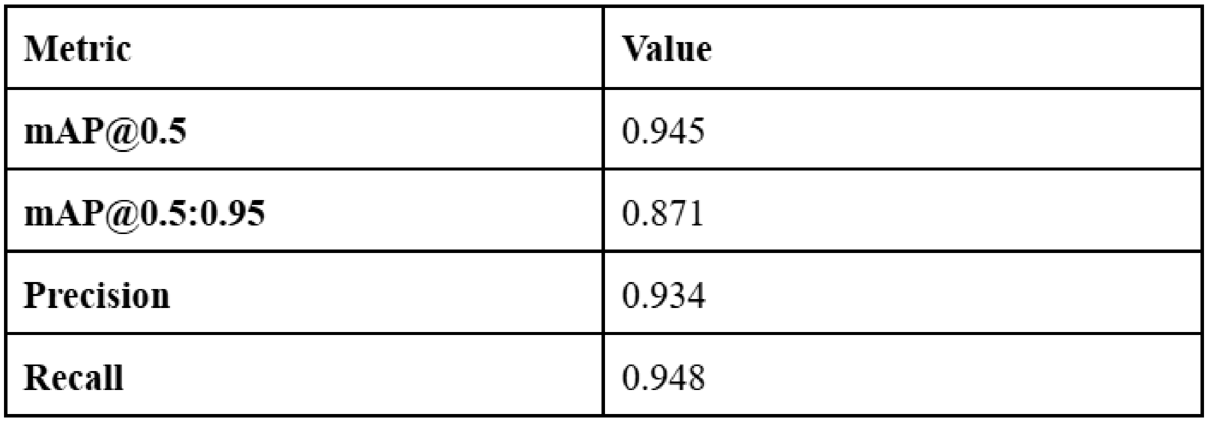
- High object detection accuracy: The model consistently achieved high accuracy in detecting and classifying objects (cars, buses, motorcycles, trucks, and ambulances).
- Low training loss: abjectness, classification, and bounding box regression losses were minimized, indicating effective learning.
- Overfitting check: The model did not exhibit signs of overfitting, as the performance gap between training and validation datasets remained small.

These results highlight that the model successfully learned the distinguishing features of vehicle classes, particularly emergency vehicles, during training.

5.4.2 Performance on Validation Data

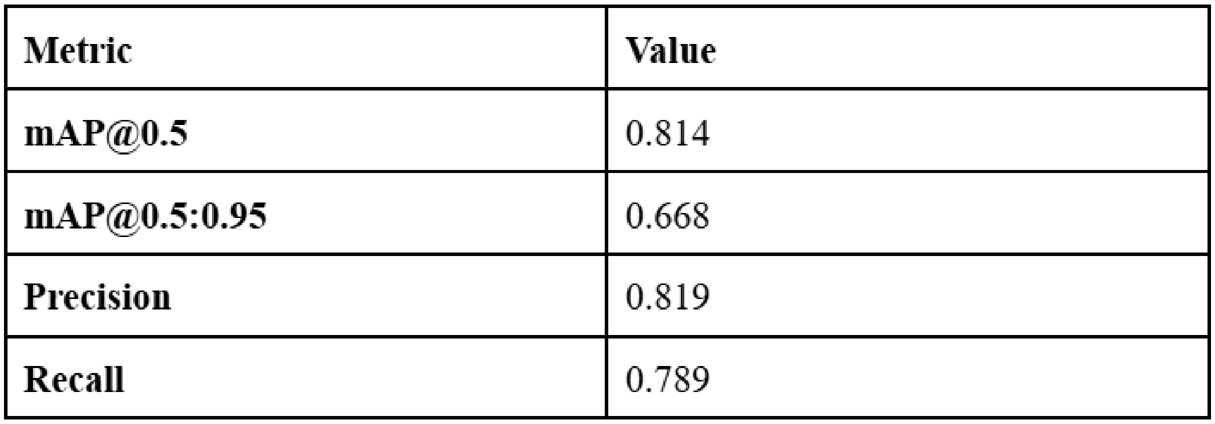
The model was validated using a separate dataset comprising real-world traffic images. The key performance metrics are summarized below, based on the screenshot of the validation output:

Vehicle & Ambulance model:



Metric	**Value**
mAP@0.5	0.945
mAP@0.5:0.95	0.871
Precision	0.934
Recall	0.948

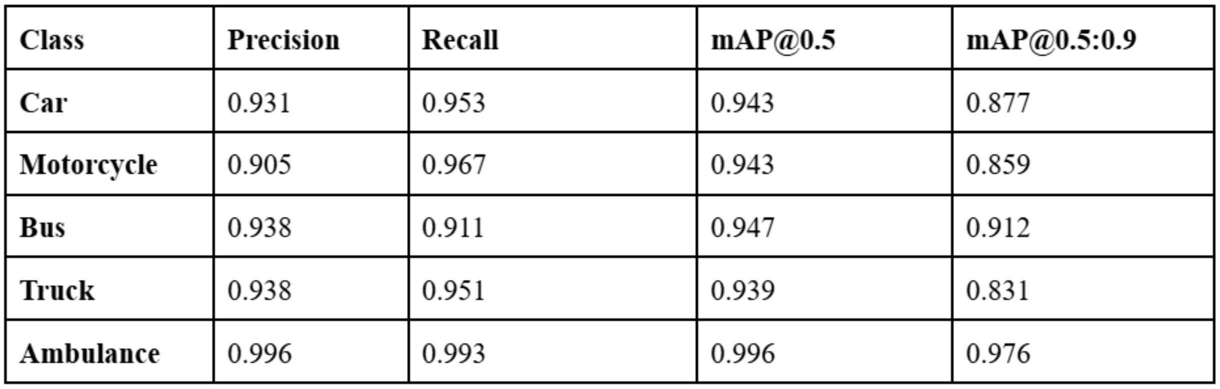
Traffic Light model:



Metric	**Value**
mAP@0.5	0.814
mAP@0.5:0.95	0.668
Precision	0.819
Recall	0.789

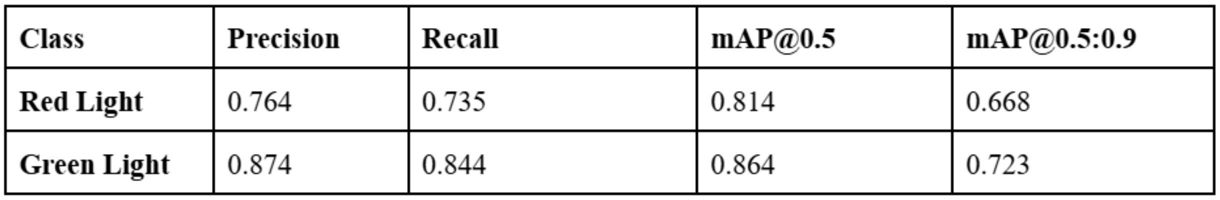
Validation Performance:

Vehicle & Ambulance model:



Class	Precision	Recall	mAP@0.5	mAP@0.5:0.9
Car	0.931	0.953	0.943	0.877
Motorcycle	0.905	0.967	0.943	0.859
Bus	0.938	0.911	0.947	0.912
Truck	0.938	0.951	0.939	0.831
Ambulance	0.996	0.993	0.996	0.976

Traffic Light model:



Class	Precision	Recall	mAP@0.5	mAP@0.5:0.9
Red Light	0.764	0.735	0.814	0.668
Green Light	0.874	0.844	0.864	0.723

The evaluation indicates that the model generalizes well from training to validation data, achieving high accuracy and robustness in detecting both standard and emergency vehicles. These results confirm the model's readiness for integration into the larger traffic monitoring and optimization system.

5.5 System Evaluation

Overall system’s efficiency beyond model accuracy. While the model may perform well in isolation, a practical deployment demands a system that is scalable, resource-efficient, and responsive under real-world conditions. The following aspects were assessed: scalability, hardware resource utilization, and system latency.

5.5.1 Scalability Assessment

Scalability refers to the system’s ability to maintain performance levels when subjected to increasing workloads, such as processing higher traffic camera feeds or running on multiple intersections simultaneously.

- Current Design: The system, built on YOLOv11 with a modular architecture, supports horizontal scaling by allowing deployment across multiple nodes or edge devices.
- Testing Scenario: The system was evaluated under simulated load by feeding simultaneous video streams.
- Result: The model maintained consistent inference performance with minimal degradation in speed and accuracy, confirming its scalability for city-wide traffic surveillance if sufficient hardware is provided.

The system demonstrates strong scalability, making it suitable for deployment in large-scale smart city infrastructures.

5.5.2 Resource Utilization

Efficient use of computational resources ensures the system remains deployable even on edge devices or constrained environments.

- Hardware Used: NVIDIA GeForce RTX 3090 GPU with 24GB VRAM.
- Utilization Metrics:
  16 core cpu
  125 gb ram
  24 gb vram
- Power Consumption: Within acceptable limits for high-performance deployments. The system is optimized for GPU-based environments and can be adapted for edge computing with slight model compression or pruning techniques, if necessary.

5.5.3 Responsiveness and Latency

Responsiveness determines how quickly the system reacts to new inputs, a crucial factor for real-time traffic monitoring and decision-making.

- Inference Time: ~1.7 milliseconds per image (as per validation log)
- Total Latency (including preprocessing and post-processing): ~2.9 milliseconds per frame
- Frame Rate Support: Capable of processing over 30 FPS in real-time scenarios

The system exhibits ultra-low latency, making it suitable for real-time applications such as traffic light control or emergency vehicle prioritization

5.6 Comparison and Discussion

Our YOLOv11-based model performed better than older models like YOLOv8 in terms of accuracy (mAP) and speed. It was able to detect different types of vehicles efficiently and quickly, which is important for real-time use. However, the model had slightly lower performance when detecting smaller objects like motorcycles, possibly due to image quality or object overlapping. Even though deep learning models can be hard to fully understand, the confidence scores and bounding boxes help us see what the model is doing. In the future, we can improve the model by adding more diverse data, using video frame sequences, and making the system work better on small devices. Overall, the results are promising, but there is still room for improvement and further research.

Chapter 6: Model Deployment and Integration

6.1 Technology Stack for AI Systems

To achieve a robust and efficient deployment of our AI-powered smart traffic management system, a carefully selected technology stack was employed. This stack was designed to accommodate the computational needs of multiple real-time deep learning models and to ensure smooth integration between the frontend user interface and backend inference systems. The following section outlines the core components of the software frameworks, libraries, and hardware infrastructure that facilitated the deployment and integration of our AI models.

6.1.1 Software Frameworks and Libraries

A variety of software tools and libraries were utilized to manage model training, real-time inference, audio signal processing, and data visualization. The core software stack includes:

- OpenCV (cv2): Utilized for image pre-processing and frame-by-frame video stream analysis, crucial for extracting frames for object detection using YOLO-based models.
- NumPy (numpy): Provided numerical operations and array manipulations essential for efficient handling of image tensors and audio spectrograms.
- PyTorch (torch): Served as the primary deep learning framework for training and deploying custom YOLO models for vehicle, ambulance, and traffic light detection tasks.

- TensorFlow (tensorflow): Used for deploying the YAMNet model, which specializes in audio event detection. TensorFlow’s efficient model loading and execution support enabled smooth real-time sound classification (e.g., detecting ambulance sirens).

- Ultralytics: This library provided cutting-edge performance for real-time vehicle detection and classification in video streams. Its ease of integration, pre-trained weights, and high inference speed made it ideal for our smart traffic monitoring needs.

- Librosa: A powerful Python library for audio analysis. It was essential in transforming audio data into mel-spectrograms and extracting relevant features to be passed into the YAMNet model.

- SoundDevice (sounddevice): Enabled real-time audio capture through connected microphones, allowing for continuous audio stream analysis and integration with the YAMNet model.

- concurrent.futures: Used to implement multithreading, which facilitated parallel model inference for video and audio data, thus ensuring minimal latency in detection and response.

- datetime and json: These libraries supported logging, formatting output responses, and managing structured data transmission between components of the system.

6.1.2 Deep Learning Models

Three models were trained, optimized, and deployed:

1. YOLO (You Only Look Once) – Used for:
   - Vehicle detection (Cars, Trucks, Motorcycles, Buses) and Ambulance recognition
   - Traffic light status detection

2. YAMNet (Yet Another MobileNet) – A TensorFlow-based audio classification model, trained and fine-tuned to recognize:
   - Siren audio signals, enabling context-aware responses (e.g., prioritizing emergency vehicles at intersections).

These models operate concurrently, and their outputs are synthesized into a meaningful decision-making framework that adjusts traffic signals and ad displays based on real-time urban traffic dynamics.

6.1.3 Integration Logic

The system’s backend logic interprets detection results to generate human-readable and actionable outputs, such as:

“

Cars detected: car count,

Trucks detected: truck count

Motorcycles detected: motorcycle count

Buses detected: bus count

Ambulances detected: True/False,

traffic_signal_command: turn red/green for {default duration} seconds,

ad_display_command: resume/pause ad,

reason: High/Moderate/Low traffic density

”

This format ensures clarity in communication between the AI backend and the frontend dashboard, which is essential for real-time monitoring and control.

6.1.4 Hardware Infrastructure

The training of deep learning models requires a high-performance computing environment. The following hardware specifications were used:

- **CPU**: 16-core processors provided parallel processing capabilities for handling multiple model inferences and data streams simultaneously.
- **RAM**: 125 GB — ensured smooth handling of large datasets and high-definition video/audio processing without memory bottlenecks.
- **GPU**: 24 GB VRAM — leveraged for real-time inference of computationally intensive YOLO and YAMNet models, drastically reducing latency and enhancing performance.

This infrastructure was crucial for ensuring that the AI system could respond to changing traffic conditions and audio signals with minimal delay, a key requirement for effective smart city applications.
By strategically combining a powerful software stack with high-performance hardware, the AI models were successfully deployed and integrated into a unified system that delivers intelligent, real-time decisions for urban traffic optimization.

6.2 Data Acquisition and Preprocessing Pipeline

To ensure the effectiveness of the deployed AI models, robust data acquisition and preprocessing pipeline were developed. This pipeline is responsible for the efficient ingestion, transformation, and preparation of both visual and auditory input data. The goal is to ensure high-quality, consistent, and compatible data across all models for accurate real-time inference.

6.2.1 Input Data Sources

The system is designed to process input from two main modalities:

- Video Streams: Captured from traffic surveillance cameras or uploaded as .mp4 files.
- Audio Signals: Extracted from the video or captured in real-time using microphones, particularly for detecting emergency sirens.

6.2.1 Region of Interest (ROI) Analysis

To ensure that only relevant activity is considered in decision-making, a Region of Interest (ROI) filtering mechanism is employed. The ROI defines a specific area within the frame where vehicle activity is analyzed—typically the lane or intersection directly influenced by a traffic signal.

A utility function **validate_roi**() ensures that the coordinates of the ROI are consistently formatted, from the top-left to the bottom-right corner, regardless of how they were initially provided. This standardization is crucial for maintaining accurate geometric computations across the pipeline.

6.2.2 Spatial Filtering and Point Validation

The is_inside_roi() function that we made determines whether a given object (based on its center point) lies within the ROI. This is particularly important for:

- Filtering out false positives or irrelevant detections outside the zone of interest.
- Determining which vehicles are waiting at the intersection or approaching a signal.

6.2.3 Preprocessing for Visual Data

The preprocessing of video input is structured around the following steps:

- Region of Interest (ROI) Extraction: To optimize computational resources and focus model predictions. This filter out irrelevant sections of the frame, ensuring that only the active road area is analyzed.
- Frame-by-Frame Analysis: Video files are read frame-by-frame using OpenCV (cv2), and each frame is passed through trained YOLOv11 models:
    - Vehicle Detection Model: Identifies and classify objects such as Car, Motorcycle, Bus, Truck, or Ambulance.
    - Traffic Light Detection Model: Recognizes traffic light states to support decision-making processes in traffic control.
- Category Mapping: Detected objects are mapped using a custom dictionary to ensure human-readable output:
  vehicle_categories = {0: "Car", 1: "Motorcycle", 2: "Bus", 3: "Truck", 4: "Ambulance"}

6.2.4 Audio Processing Pipeline

The audio preprocessing pipeline is particularly focused on siren detection using a custom-trained YAMNet model. The pipeline follows these key stages:

- Audio Extraction: If audio is embedded within a video file, it is extracted using moviepy, to ensure audio compatibility and proper formatting before it is processed by the model.
- Resampling and Duration Capping: Using librosa, the waveform is resampled to 16 kHz and capped at 10 seconds to reduce noise and irrelevant data.

- Tensor Conversion and Inference: The waveform is converted into a TensorFlow tensor and passed to the YAMNet model; Siren detection is confirmed if any score from the relevant class exceeds a defined threshold.

6.2.5 Data Quality and Compatibility

To ensure high data integrity and consistency across modalities, several strategies were adopted:

- Unified Sample Rate: All audio inputs are standardized to 16 kHz for consistency with the YAMNet model requirements.
- Thresholding and Score Filtering: Scores below the detection confidence threshold are filtered out to minimize false positives.
- Frame Synchronization: Video and audio data are processed in a synchronized manner to maintain temporal accuracy in real-time decision making.
- Temporary File Handling: Temporary paths are created using tempfile to prevent filesystem conflicts during dynamic file generation and processing.

This preprocessing pipeline ensures that all incoming data, regardless of its source, is transformed into a model-ready format that is both computationally efficient and aligned with the trained AI models' expectations. By enforcing strict quality control and preprocessing standards, the system maintains high reliability and real-time responsiveness in diverse traffic conditions.

6.3 Decision-Making System

Following the data acquisition and preprocessing stage, the decision-making system plays a critical role in interpreting the output from the detection models to inform actionable insights. This subsystem processes spatial and categorical information derived from the YOLO detection models to assess traffic conditions and guide dynamic control mechanisms, such as prioritizing emergency vehicles or adjusting traffic signal timing.

6.3.1 Vehicle Density Calculation

A key metric used in the system's logic is vehicle density, which reflects the ratio of detected vehicles inside the ROI to the total number of detected vehicles in the frame. The calculate_density() function is responsible for computing this value. The process involves:

- Counting all detected vehicle objects.
- Identifying which of those objects have center points falling within the defined ROI.
- Calculating the density as a fraction, ranging from 0.0 (no vehicles in ROI) to 1.0 (all vehicles in ROI).

This density value serves as a quantitative indicator of traffic congestion at a specific segment of the road. When used in conjunction with other factors such as traffic light status and emergency siren detection, it enables the system to: Trigger signal timing adjustments, prioritize green light phases for lanes with high vehicle density and detect the presence of emergency vehicles (e.g., ambulances with sirens) and provide them right way when needed.

6.3.2 Integration with Decision Logic

The outputs from the spatial and density computations are integrated into a broader rule-based or model-driven logic layer. This layer is responsible for making final decisions based on:

- Vehicle type and count
- Presence of emergency vehicles (based on audio classification)
- Current traffic light phase
- Time-of-day patterns

This decision-making system enables responsive and context-aware behavior that enhances road efficiency and safety—especially in urban environments with fluctuating traffic patterns.

6.4 Model Deployment Strategies

The deployment phase represents a critical bridge between model development and real-world application. In this project, the integration and deployment of deep learning models were meticulously engineered to ensure reliability, scalability, and responsiveness in real-time traffic scenarios. This section discusses the deployment strategies and architectural considerations undertaken to operationalize the trained models effectively within the web-based system.

6.4.1 Deployment Architecture

The system architecture is designed around a multi-modal, multi-model inference pipeline that seamlessly combines both audio and video streams. The input layer accepts video uploads or real-time feeds from users, along with audio data where applicable. These inputs are processed concurrently by three core deep learning models.

These models operate in parallel and feed their output into a unified Model Integration Pipeline. This integration layer functions as a communication interface, aggregating and synchronizing multimodal data to ensure consistent input for the downstream Decision-Making System (as outlined in Section 6.3). The final outcomes control two key systems:

- Traffic Signal Control, where light phases are dynamically adjusted based on congestion and emergency detection.
- Advisory Display Control, which can alert drivers and pedestrians about nearby ambulances or unusual traffic events.

6.4.2 Deployment Methodologies

For the deployment of the system, lightweight and flexible architecture was adopted using **Flask** as the web framework. Flask was chosen for its simplicity and seamless integration with Python-based deep learning models. This framework serves as the bridge between the backend model inference logic and the user interface.

The **front end** of the application was developed using **HTML, CSS, and JavaScript**, enabling a clean and responsive user experience. Through this interface, users can upload video files or access live camera feeds, view traffic status updates, and receive emergency vehicle alerts in real time. JavaScript handles dynamic content updates and asynchronous interactions, allowing real-time communication with Flask endpoints without requiring full page reloads.

The Flask backend hosts multiple routes for handling tasks such as:

- Uploading and processing video/audio data.
- Serving inference results from the deployed deep learning models.
- Returning dynamic alerts and visualization data to the frontend.

Each model (vehicle detection, traffic light detection, and siren detection) is encapsulated as a callable module within the Flask application. Once the input data is received, these modules process it and return structured outputs to the interface, making the system highly interactive and user-friendly.

This stack ensures maintainability and extensibility, making it well-suited for both academic experimentation and real-world deployment scenarios in smart traffic systems.

6.4.3 User Experience and Trial Deployment

The website offers users the ability to upload traffic videos or stream live camera feeds. The backend processes these inputs through the pipeline and returns visual and textual insights in near real-time. These outputs are then displayed in a user-friendly format, highlighting detected vehicles, emergency sirens, and traffic signal states, alongside system-generated decisions (e.g., whether to prioritize emergency passage).

A deployment phase was conducted to validate system performance. This trial helped refine interface elements, adjust model thresholds, and evaluate the robustness of the integration pipeline.

6.4.4 Key Features and Future Enhancements

The deployed system offers the following core functionalities:

- **Vehicle Detection and Classification**
  Utilizes computer vision models to identify and categorize different types of vehicles in real time.

- **Emergency Vehicle Detection and Prioritization**
  Detects ambulances and other emergency vehicles and prioritizes their movement by adjusting signal phases accordingly.

- **Traffic Light Status Monitoring**
  Recognizes and interprets the current status (Red, Green) of traffic signals to support decision-making.
- **Contextual Advertising System**
  Displays targeted advertisements based on the traffic light status, utilizing idle vehicle time during red signals.
- **Web-Based Visual Analytics Interface**
  Provides an interactive and user-friendly dashboard for monitoring traffic flow, signal timing, and alert notifications.

Future developments may include:

To further improve system intelligence and integration, the following developments are proposed:

- **IoT Device Integration**
  Incorporating sensors and smart cameras to enhance real-time data acquisition and system responsiveness.
- **Accident Detection Capabilities**
  Extending the current detection pipeline to recognize collisions or abnormal traffic events using spatio-temporal patterns.
- **Automated Traffic Signal Control**
  Enabling direct API-based communication with traffic light controllers for fully automated signal optimization.
- **Reinforcement Learning-Based Optimization**
  Replacing rule-based control logic with a reinforcement learning (RL) agent trained to maximize traffic efficiency through adaptive decision-making.

6.5 Model Integration within AI Systems

The vehicle detection and classification model are seamlessly integrated into the overall AI-driven traffic monitoring and control system. This integration ensures real-time responsiveness, scalability, and modularity across all components. The following outlines the key aspects of model integration:

1. Modular Architecture: The AI model is deployed as a modular component within the backend system. It communicates with other modules (e.g., ROI detection, traffic light status module, emergency vehicle identification, and advertising logic), allowing for easy updates and enhancements without disrupting the full system.

2. Real-Time Data Flow: Live video feeds from video files are continuously streamed into the system. These streams are processed frame-by-frame using the detection model, which identifies and classifies vehicles, extracts bounding boxes, and returns metadata such as vehicle type, position, and confidence level.

3. Interaction with Decision-Making Modules: The output from the AI model serves as critical input to the system's decision-making layers. For example:

- The vehicle positions are cross-referenced with the Region of Interest (ROI) module to calculate traffic density.
- Detected emergency vehicles trigger prioritization rules.
- Detected vehicle counts influence advertising logic and future signal control strategies.

4. Communication with Frontend and Visualization

All processed results are communicated to the frontend via Flask APIs. The data is presented as interactive and user-friendly with the built in HTML, CSS, and JavaScript, allowing users to interactively view analytics such as current traffic conditions, density charts, and vehicle classifications.

5. Performance Optimization

To maintain performance, the model is optimized for batch processing and hardware acceleration (e.g., GPU/CPU inference), ensuring minimal latency in detection and decision-making even under heavy load.

Chapter 7: Conclusion and Future Work

This chapter provides a comprehensive overview of the Smart Traffic Management System's development, implementation, and performance. It encapsulates the project's objectives, achievements, and key takeaways, and proposes potential directions for future development. By reflecting on the lessons learned and the system's current capabilities, this chapter lays the foundation for enhancements that can be explored beyond the scope of this initial prototype.

7.1 Summary and Project Evaluation

The Smart Traffic Management System was designed with the primary objective of alleviating traffic congestion, enhancing emergency vehicle prioritization, and laying the groundwork for intelligent, adaptive traffic control. To achieve this, the project integrated multiple machines learning components, including computer vision models and audio analysis pipelines, to process live or recorded traffic data.

Key accomplishments of the project include:

- **Object Detection and Classification**: A custom-trained YOLOv11 model was employed to detect and classify vehicles into categories such as car, truck, bus, motorcycles, and specifically ambulances. The system demonstrated high accuracy and reliability under varied lighting and environmental conditions.
- **Traffic Light Recognition**: A separate YOLOv11-based model was used to detect traffic lights. OpenCV post-processing was integrated to accurately determine the color state (red or green) of each detected light, allowing the system to understand and react to signal conditions.

- **Siren Detection via Audio**: The system processed audio extracted from video files to detect emergency sirens using a dedicated siren classification model trained on labeled spectrogram images. This provided an additional verification layer for ambulance detection, improving reliability in noisy or visually obstructed environments.
- **Central Orchestrator Pipeline**: A modular pipeline was developed to handle MP4 inputs or camera inputs, separating and synchronizing video and audio streams. Each stream was routed to specialized models, and outputs were returned in a standardized dictionary format for integration into the decision engine.
- **Traffic Decision Engine**: Based on model outputs, the engine makes real-time decisions, such as giving priority to ambulances, adjusting traffic signals to reduce congestion, and ensuring safe and efficient traffic flow.

The project achieved its intended goals and demonstrated the feasibility of using AI and computer vision in smart traffic control. The modular and scalable design allows for easy integration of future enhancements. While the system is currently in prototype phase and not connected to live camera feeds or real-world traffic lights, it sets a solid foundation for real-world deployment.

7.2 Future Directions

1. IoT Integration

Integrating IoT (Internet of Things) devices into the system can significantly expand its real-time capabilities. By embedding smart sensors in traffic lights, intersections, and emergency vehicles, the system can:

- Receive real-time updates from physical traffic signals.
- Send adaptive control commands to IoT-enabled traffic lights based on live model decisions.
- Track emergency vehicles using GPS and vehicle-to-infrastructure (V2I) communication.
- Detects traffic density using real-time road sensors, enhancing decision accuracy.

This would allow the prototype to evolve into a real-time smart infrastructure capable of live monitoring and dynamic signal control.

2. Accident Detection and Response

Future versions of the system can be expanded to include accident detection using visual cues and audio anomalies. A dedicated accident detection module could:

- Trigger automatic alerts to emergency services.
- Reroute traffic around the incident area.
- Log accident data for city analytics and infrastructure improvement. Integrating this feature would further improve urban safety and emergency response times.

3. Automated Traffic Signal Control

Currently, the decision engine simulates signal control actions. In the future, the system can be connected directly to smart traffic lights for automated control. This real-time actuation could be achieved through:

- APIs provided by city traffic departments.
- Custom-built IoT relays on signal controllers.
- A centralized control panel to manage city-wide intersections dynamically.

This automation would allow the system to directly optimize traffic flow, reduce idle time, and prioritize vehicles such as ambulances and public transportation.

4. Reinforcement Learning for Continuous Optimization

A promising future enhancement is the incorporation of reinforcement learning (RL) to improve the decision-making engine over time. Using feedback loops and performance metrics (e.g., reduced congestion, shorter ambulance response time), the RL agent can learn and adapt traffic control strategies based on past outcomes.

- The reward function could be based on traffic clearance time, ambulance priority effectiveness, and accident avoidance.
- Simulation environments or digital twins of intersections could be used to train RL models safely before real-world deployment.
- The system could adapt to seasonal patterns, time-of-day variations, and unpredictable events (e.g., construction or public events).
- Data Storage for Retraining: All processed data, including frames and metadata, was stored in MongoDB for potential future model retraining and analytics.

5. Cloud Deployment and Scalability

To enable wide-scale deployment, the system can be hosted on cloud infrastructure, allowing multiple intersections and areas to be managed simultaneously. This would involve:

- Deploying the pipeline as microservices.
- Scaling inference engines using GPU-backed servers or edge devices.
- Exposing secure REST APIs for cities to interface with the system.

References

Defining Smart Traffic Management Systems
ResearchGate STM
Amman Smart City Roadmap
Technology Definition, Limitations & Disadvantages - Lesson | Study.com
Autonomous Smart Traffic Management System Using Artificial Intelligence
The official website for the City of London
Singapore Traffic Management System
ATSAC
LADOT
Seoul Topis
https://www.europarl.europa.eu/RegData/etudes/BRIE/2019/640163/EPRS_BRI(2019)640163_EN.pdf