## Final Year Project

# MSc Artificial Intelligence & Machine Learning

### Accident Detection in Urban Traffic

**OMAR TAREK ELSAYED MASOUD**

**2270587**

**Project unit:** Master's Project
**Supervisor:** Dr. Gelayol Golcarenarenji

**17 September 2025**

# Table of Contents

# 1 Introduction

## 1.1 Context of Research

Road accidents are statistically one of the most serious problems worldwide, as they threaten public health, as well as the socioeconomic stability. According to the World Health Organisation, road accidents take the lives of approximately 1.35 million people annually, and on top of that tens of millions of people suffer great injury or become disabled (WHO,2018). There are more issues with it other than the great human cost such as accidents deprive most countries of an estimated 3% of their GDP . The steady rise in urbanization and vehicle ownership will continue to elongate the scale of the problem unless serious measures start to be implemented.

The closed-circuit television (CCTV) networks and electronic surveillance are at the heart of traffic control methods in modern urban cities. These networks are installed by default on highways, intersection points, and any traffic area prone to accidents, this is done to promote safety and assist law enforcement. Yet, they are operated with human operators, and they are responsible for monitoring these systems with dozens of video streams (Chen et al., 2020). This presents two problems: accidents could happen and remain unseen due to operator fatigue, and slow response times when accidents happen which causes traffic congestion and secondary collisions. The scale of city traffic data in real time is too large to be monitored by humans.

Transportation systems are increasingly leaning on computer vision and artificial intelligence (AI) technology to overcome these challenges. AI-powered systems exponentially surpass human capabilities when it comes to processing video data, recognizing abnormalities in traffic patterns, and reporting possible accidents. Computer vision, in particular, enables us to extract useful information directly from unprocessed video, enabling the tracking and analysis of traffic flow, pedestrians, and vehicles without intrusive hardware deployments (Redmon et al., 2016; Ultralytics, 2023). Intelligent Transportation Systems (ITS), a field of study that seeks to make transportation infrastructure safer, more efficient, and responsive to changing needs, is based on the integration of several technologies.

Deep learning has driven a revolution in computer vision, with convolutional neural networks (CNNs) providing robust feature extraction from images and videos. More recently, object detection models such as the YOLO (You Only Look Once) family have advanced the state of the art by combining high detection accuracy with near real-time inference (Bochkovskiy et al., 2020). These models have been applied to diverse traffic-related tasks, including vehicle counting, traffic light detection, and pedestrian recognition, making them strong candidates for accident detection. Despite these advances, challenges remain. Accidents are relatively rare events compared to normal traffic, resulting in highly imbalanced datasets that bias models towards negative (non-accident) classifications (Gupta et al., 2021). Furthermore, environmental conditions such as rain, night-time lighting, and occlusions from other vehicles often reduce detection accuracy, highlighting the need for robust preprocessing and model training strategies.

Accident detection differs fundamentally from other traffic tasks such as congestion analysis or traffic flow prediction because it demands high recall. Missing even a small number of accidents can result in delayed emergency responses and loss of life. At the same time, overly sensitive systems risk overwhelming operators with false positives, which may reduce trust and usability. The balance between precision and recall is therefore a critical design consideration. For practical deployment, accident detection systems must also operate with low latency and computational efficiency, particularly in cities where hundreds of video feeds may need to be processed in parallel.

This project is situated within this context, addressing the intersection of AI, computer vision, and intelligent transportation systems. The research proposes a hybrid framework that combines CNN-based scene classification with YOLOv8 object detection. The CNN acts as a lightweight first-stage filter, identifying candidate accident frames while conserving computational resources. YOLOv8, a state-of-the-art anchor-free detector, is then applied to localise accident cues with bounding boxes, providing interpretable evidence for human operators. By integrating these two models, the system seeks to strike a balance between efficiency and accuracy, making it both practical for large-scale deployment and reliable in safety-critical scenarios.

## 1.2 Project Aim

For this project, the aim is to design, implement, and evaluate a hybrid accident detection system that uses Convolutional Neural Networks (CNNs) with YOLO-based object detection models (YOLOv8 and YOLOv11). This system is intended to operate on CCTV traffic footage in real time to detect accidents with high recall while also making sure it is computationally efficient. A major objective is to enhance explainability through explainable AI methods like Grad-CAM and bounding box visualizations, which makes sure that the output is transparent and trustworthy for humans. This project aims to contribute to the development of practical, accurate, and ethically responsible accident detection systems for smart city environments by facing common challenges like dataset imbalance, low-light conditions, and the need for scalable deployment.

## 1.3 Project Objectives

To achieve the aim of this project, several objectives have been established. The work begins with a comprehensive literature review of existing AI-based methods for traffic accident detection, focusing on CNN and YOLO models in order to identify their strengths, limitations, and research gaps. Following this, a large-scale CCTV accident dataset will be collected and prepared, with attention given to issues such as class imbalance, variation in environmental conditions, and the ethical use of surveillance data. The modelling phase involves implementing and training a CNN classifier for binary scene-level accident detection, alongside fine-tuning YOLOv8 and YOLOv11 object detectors to enable spatial

localisation of accident cues. These models will then be integrated into a hybrid pipeline designed to combine computational efficiency with accurate detection.

To ensure explainability, methods such as Grad-CAM and EigenCAM for CNN predictions and bounding box overlays for YOLO detection will be integrated. The system will be heavily tested using metrics including accuracy, precision, recall, F1-score, mean Average Precision (mAP), latency, and frames per second (FPS), with high recall emphasized specifically. Comparisons will be made between YOLOv8 and YOLOv11 results, and the value added by the hybrid approach. Finally, the project will address ethical and professional considerations, e.g., privacy, fairness, reproducibility, and deployment challenges in the real world, and propose areas for improvement such as stronger dataset annotation, addition of temporal tracking, and more optimisation for large-scale deployment.

# 2 Literature Review (Critical Review)

## 2.2 Research Background and Definitions

This project applies deep learning and computer vision techniques to detect accidents in urban traffic areas, the specific techniques that will be used are YOLO based models. For context, it is important to define multiple key terms and models that will be used throughout the project.

- **Object Detection -** Object Detection is the task of identifying and classifying objects such as vehicles, pedestrians, or traffic signs in an image or a video. For this project, object detection is done using **YOLO (You Only Look Once)** structure, which is a real time object detection algorithm that is well known for its speed and accuracy. There are multiple versions of it and among them, **YOLOv4, YOLOv5, and YOLOv8** are commonly used in traffic management due to their high frame per second (FPS) capabilities and efficient box prediction.

- **Accident Detection** is a specific use of object detection and motion detection where in events such as collisions, sudden braking, or vehicle rollovers are detected through visual features. Moreover, this is usually supported by a tracking algorithm such as **Kalman Filters** and **DeepSORT,** which track the movements of objects across video frames and help in identifying behaviours that are abnormal to the behaviour of traffic.

- **Computer Vision** is a branch of Artificial Intelligence that allows for systems to interpret and act on visual data. For the world of traffic management, it enables non-intrusive monitoring through CCTV feeds, which allows for vehicle classification, estimating the length of the queue, and detecting incidents all without the need for any physical sensors.

- **Supervised Learning** is a branch of Machine Learning where models are trained on labelled datasets. This project relies a lot on supervised learning because accident detection models do require annotated images showing the vehicle types, positions, and accident states to learn effectively. Supervised models like YOLO are preferred in this context as they generalize well on real world video and the outputs are transparent and interpretable.

- **Frame rate (FPS)** and **latency** are important real time metrics. In traffic video analysis, a model should process video streams at high FPS (usually around 25-30 or more) and low latency (200 milliseconds and under) to be able to detect events as they occur. YOLO models, particularly YOLOv5 and YOLOv8n, are optimized for these things.

- **Intersection monitoring** is a crucial element of smart traffic systems. Intersections are a hotspot for heavy traffic and accidents, managing intersections requires accurate detection of vehicles and pedestrians across many lanes and different directions. This project would focus on detecting incidents at intersections, where quick decision-making can greatly improve safety and the traffic flow.

- **Traffic anomalies** refer to behaviours that branch off from the expected patters such as reverse driving, suddenly stopping, or unusual density buildup. The first step on identifying potential accidents or system malfunctions is to detect these anomalies.

- **Convolutional Neural Networks (CNNs) –** CNNs are a class of deep neural networks designed to process visual data. They use convolutional layers to automatically extract spatial features such as edges, textures, and object parts, which are then combined through pooling and fully connected layers for classification (LeCun et al., 1998; Krizhevsky et al., 2012). CNNs have been widely adopted for image recognition tasks due to their ability to learn hierarchical representations directly from raw pixel data. In this project, the CNN serves as a lightweight first-stage classifier to distinguish between accident and non-accident scenes, filtering frames efficiently before passing potential accidents to more complex detection models.

- **YOLOv8** is one of the latest and most widely used real-time object detection models developed by Ultralytics. Unlike earlier anchor-based versions, YOLOv8 adopts an anchor-free detection head and a C2f backbone structure, enabling higher accuracy and efficiency in detecting objects across varied scales (Jocher et al., 2023). It offers strong performance across multiple domains, maintaining both low latency and high precision, making it well-suited for real-time urban traffic monitoring. In this project, YOLOv8 is applied to localise accident cues in CCTV images, drawing bounding boxes that provide interpretable evidence for operators.

- **YOLOv11** is the newest release in the YOLO family, building upon YOLOv8 with an improved backbone and transformer-based modules for feature extraction

(Ultralytics, 2024). It delivers enhanced robustness in challenging environments, such as low-light, occluded, or high-density traffic conditions. YOLOv11 also introduces optimisations in speed and memory efficiency, making it promising for deployment at scale. This project compares YOLOv11 with YOLOv8 to assess whether its architectural advancements yield measurable improvements in accident detection accuracy and reliability.

- **Explainability (Grad-CAM and EigenCAM)** – Explainable AI methods provide visual interpretations of model decisions, making outputs more transparent to end-users. Gradient-weighted Class Activation Mapping (Grad-CAM) highlights the regions of an image that most influenced a CNN's classification, producing heatmaps that improve operator trust and accountability (Selvaraju et al., 2017). EigenCAM, an alternative, generates smoother and less noisy activation maps by leveraging principal component analysis on feature activations (Muhammad & Yeasin, 2020). In this project, these explainability tools are used to provide insights into why frames are classified as accidents, ensuring the system is not a "black box" but instead delivers evidence-based outputs.

- **Evaluation Metrics** – A robust evaluation requires multiple metrics to capture both classification and detection performance. Accuracy, precision, recall, and F1-score are used to assess the CNN classifier, with particular emphasis on recall due to the safety-critical nature of accident detection (Sokolova & Lapalme, 2009). For object detection models (YOLOv8 and YOLOv11), mean Average Precision (mAP@0.5 and mAP@0.5:0.95) quantifies detection quality across Intersection over Union thresholds (Everingham et al., 2010). In addition, latency and frames per second (FPS) are measured to verify that the models operate effectively in real-time traffic surveillance environments.

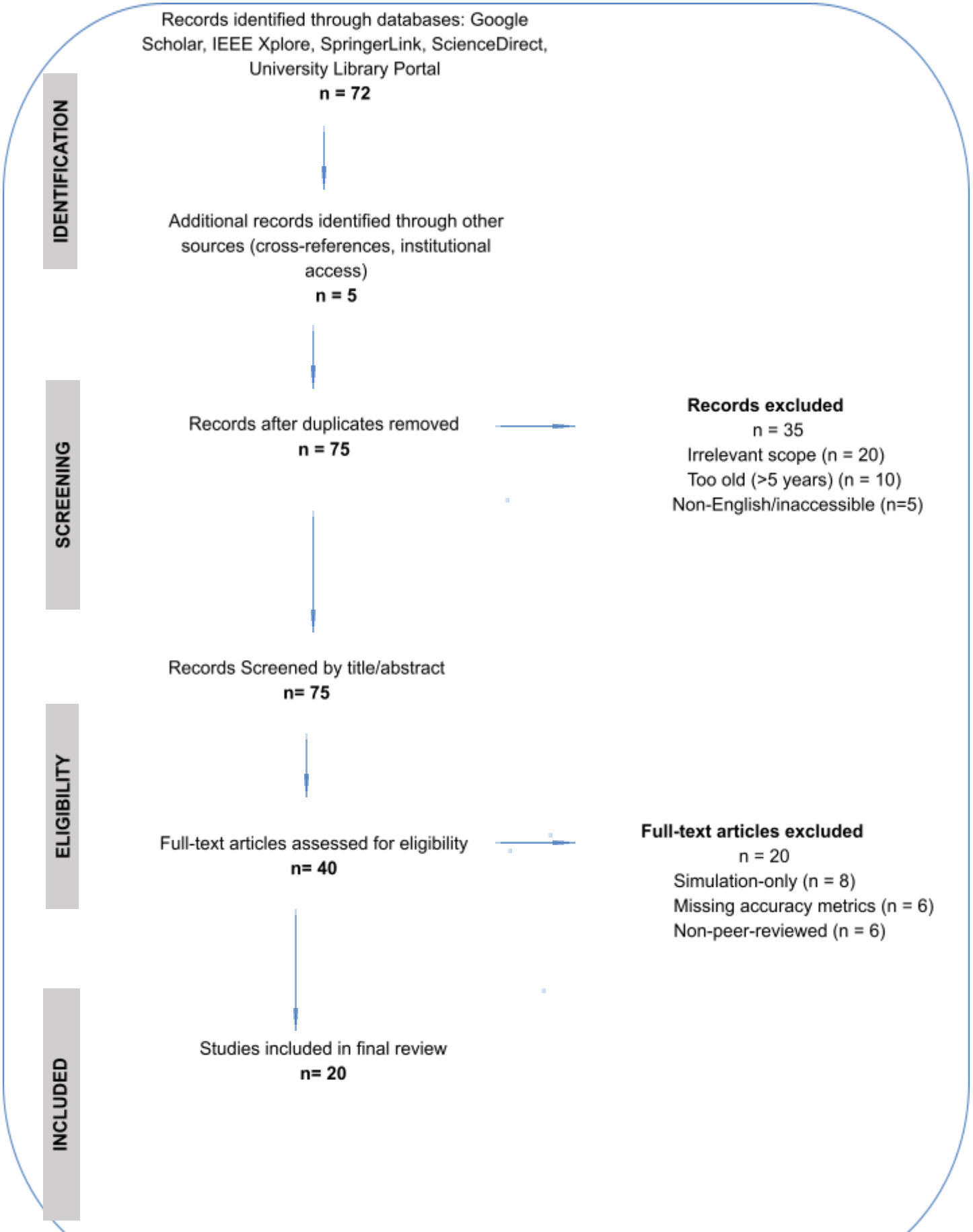



**Figure . PRISMA flow diagram illustrating the identification, screening, eligibility, and inclusion process, which resulted in 20 studies being selected for the critical review.**

## 2.3 Similar Works

In search for similar relevant studies, multiple academic databases and digital libraries were utilized. These included Google Scholar, IEEE Xplore, ScienceDirect, SpringerLink, and the University of Portsmouth Library Portal. Broad search terms were used first such as “traffic management using AI” and “computer vision for traffic control”. As the project’s focus narrowed, I refined my queries to more specific terms including “YOLO traffic accident detection”, “YOLOv5 vehicle tracking”, “real-time traffic surveillance deep learning”, and “urban intersection anomaly detection”. The strategy used in the search developed over time to capture both visual detection models and control strategies, ensuring that approaches that are relevant to both supervised learning and real time deployment scenarios are included. This refinement was necessary because the initial broad terms mostly returned reinforcement learning studies focused on traffic signal control, which were outside the scope of accident detection. Narrowing down to YOLO-based and CNN-based keywords ensured that the final 20 studies were directly relevant to computer vision approaches for real-time accident detection.

After screening and scrolling over 75 pages, inclusion criteria was applied based on:
- Recency (published within the last 5 years)
- Practical application to traffic management or accident detection
- Use of deep learning, preferably CNN or YOLO-based
- Availability of accuracy metrics or real-world evaluation

After this filtering process, 20 primary studies have been selected which are presented in the literature review table (section 2.2). The following table is a summary to the selection process in line with the PRISMA framework:

| **Phase** | **Number of Papers** |
|---|---|
| Initial search results | 75+ |
| After title/abstract screening | 40 |
| After full-text assessment | 25 |
| Final papers included | 20 |

The selected studies fall into three major categories: YOLO-based accident detection, reinforcement learning for signal control, and hybrid CV models for traffic analysis. For example, [1] used YOLOv4 with Kalman Filters to detect crashes with high accuracy in real-world datasets, while [10] demonstrated how YOLOv5 and DeepSORT can be integrated for severity classification of incidents. Meanwhile, RL-based frameworks like those reviewed in [17] discussed promising results in signal optimization but often lacked integration with real-time video analysis, limiting their applicability for this project’s scope.

Critically, while many of the reviewed works reported high detection accuracy (90–99%), several limitations emerged. A common issue was scalability, many models were evaluated

on isolated intersections or synthetic data, making generalization difficult. Others, such as [8] and [16], were GPU-dependent or required annotated datasets that are not always available. Despite this, the YOLO family of models stood out for their balance of speed, accuracy, and modularity, especially when combined with lightweight tracking mechanisms.

Based on this review, it was concluded that a YOLO-based supervised learning pipeline would be the most effective and practical foundation for building a real-time accident detection system. Its strong track record in urban surveillance tasks and its compatibility with common traffic datasets directly influenced both the design and implementation choices of this project. The reviewed literature emphasized the need for models that offer low-latency inference, strong generalization to real-world conditions, and integration capabilities with temporal tracking components. As a result, the planned system architecture will include a pre-trained YOLO detection module paired with a tracking algorithm (e.g., DeepSORT or Kalman Filter) to monitor object movement across frames and infer collision events. Additionally, insights from reviewed works highlighted the importance of modular design, enabling each component—detection, tracking, logging, and alerting—to be developed and tested independently. Several studies also discussed post-processing strategies for reducing false positives, which will be considered when implementing the decision logic around alert triggers.

By synthesizing these contributions, the project aligns itself with the most promising and practical direction in the field: leveraging high-speed, high-accuracy detection with real-world viability, while incorporating lessons learned from past implementation challenges. These findings directly feed into the requirement specifications, model selection strategy, and evaluation metrics defined in the methodology.

| Ref | Algorithm(s) | Platform | Accuracy (%) | Dataset used | Contribution(s) | Limitations |
|---|---|---|---|---|---|---|
| 1 | YOLOv4 + Kalman Filter | Python | 95% | Real-world accident dataset | Crash detection & tracking | Limited scalability |
| 2 | YOLO + CNN | Python, OpenCV, Keras | 92.3% | Intersection video | Real-time adaptive signal switching | Small-scale only |
| 3 | YOLOv7 | Python, OpenCV | 91% | Simulated & field data | High-speed object detection | No control integratic |
| 4 | YOLOv5 Ensemble | Python, SUMO, OpenCV | 90.25% | Real + synthetic traffic | Multi-type vehicle detection | GPU dependent |
| 5 | YOLOv4 | Python, OpenCV | 88% | Real intersection images | Low-latency adaptive switching | Prototype scale only |
| 6 | Faster R-CNN + Inception-v2 | TensorFlow | 99% | Urban sign images | High-accuracy sign recognition | Sign-focused only |
| 7 | Faster R-CNN + Active Nets | PyTorch | 88% | Traffic video | Occlusion-resistant segmentation | Not pipeline-tested |
| 8 | DL (XAI, Unity) | Unity, Python | 91.8% | Simulation | Explainable sign recognition | No real-world valida |
| 9 | YOLOv3 | OpenCV | 93% | CCTV crash video | Accident detection (rollover, rear-end) | Precision-focused |
| 10 | YOLOv8 + DeepSORT | Python, OpenCV | 99.2% | Traffic footage | Severity-based classification | Visual only |
| 11 | Traffic-ConvLSTM | Python | N/A | Wuhan traffic data | City-scale anomaly detection | Weather-sensitivity |
| 12 | Vision + Decision Tree | Python | N/A | CCTV traffic feeds | Accident classification | Model interpretabilit |
| 13 | CNN-LSTM | Python | >92% | Hong Kong sensor data | Flow prediction with clustering | Sensor error sensiti |
| 14 | DSGCN | Python | N/A | Grid-based traffic maps | Congestion forecasting | Training cost |
| 15 | CNN/SVM/ANN | Python | ~89% | Lima intersection | Pedestrian-vehicle flow optimization | Fixed scenario |
| 16 | CNN + Attention LSTM | Python | 97% | IoT-enabled crosswalks | Smart congestion control | IoT infrastructure ne |
| 17 | Review: ML for Traffic | Mixed | N/A | N/A | Survey of ML/DL in traffic | Not experimental |
| 18 | Autonomous Flow Agent | Python | N/A | Sensor-based flow | Time-series anomaly detection | No visual input |
| 19 | YOLOv5-based ATMS | Python | N/A | Urban camera footage | Vision-guided signal adjustment | Edge performance limits |
| 20 | ResNet (Jam Detection) | Python | >95% | Camera images | Jam classification | Not integrated |

## Limitations for each reviewed study

Limited scalability; tested on few feeds with scene-specific tuning, no validation on diverse intersections.

**[2] Sindagi et al., 2025 — YOLO + CNN**
Small-scale (single intersection) with no cross-scene transfer or control-policy learning, restricting generalisation.

**[3] MDPI Sensors, 2023 — YOLOv7**
Good accuracy, but lacks integration with control/alerting and continuous CCTV validation. Most tests were on controlled trials.

**[4] Dirit & Ahmed, 2024 — YOLOv5 Ensemble**
Accuracy improved but GPU-heavy and reliant on synthetic data, raising concerns over real-time deployment and domain transfer.

**[5] Scienceopen, 2022 — YOLOv4**
Prototype study, not validated over long periods or multiple sites. Maintainability and re-training challenges unaddressed.

**[6] Chaudhuri, 2023 — Faster R-CNN + Inception-v2**
Accurate for signs but not accident cues; narrow scope limits applicability to accident pipelines.

**[7] Arif et al., 2024 — Faster R-CNN + Active Nets**
Occlusion handling promising, but computationally heavy with no pilot real-time testing.

**[8] Smajic et al., 2021 — DLXAI in Unity**
Simulation-only with no CCTV transferability. Lighting and camera stability remain unresolved.

**[9] IJSLR, 2023 — YOLOv3**
Focused on few crash types; dataset small, city-scale deployment concerns not addressed.

**[10] Ahmed et al., 2023 — YOLOv8 + DeepSORT**
Severity labels based on visuals only, prone to misclassification without kinematic/context data.

**[11] Xu et al., 2023 — Traffic-ConvLSTM**
Detects anomalies but unstable under varied conditions. Not optimised for frame-level accident localisation.

**[12] Saleh et al., 2022 — Vision + Decision Tree**
Interpretable but lacks spatial evidence and multi-site testing; not robust for complex accident detection.

**[13] Gao et al., 2024 — CNN-LSTM**
Strong flow prediction (>92%) but sensor-dependent, with no CCTV accident validation.

**[14] Chen et al., 2022 — DSCGN**
Accurate forecasts but high training cost and reliance on curated grids limit generalisation.

**[15] Quispe et al., 2023 — CNN/SVM/ANN**
Evaluated only in limited settings. Hybrid ML stack raises overhead with no real-world validation.

**[16] Rani & Ramasamy, 2024 — CNN + Attention LSTM**
High reported accuracy (~97%), but IoT reliance constrains deployment. Accident localisation not central.

**[17] Ghosh & Basu, 2024 — Survey (ML/DL for traffic)**
Taxonomy-focused, not experimental. Useful overview but no metrics or benchmark validation.

**[18] Karaimi et al., 2023 — Autonomous Flow Agent**
Sensor-based, no visual input; transfer to camera streams would need redesign.

**[19] Patel et al., 2024 — YOLOv5 ATMS**
Promising, but edge computing limits throughput; multi-stream scalability untested.

**[20] ResNet (Jam Detection)**
High jam classification (>85%), but no integration with alerting or traffic management pipelines.

## 2.4 Reports Explanation

[1] YOLOv4 + Kalman Filter

Combined YOLOv4 with Kalman Filters for accident and vehicle tracking. Achieved strong detection accuracy and smooth tracking, but struggled with scalability in complex intersections.

[2] YOLO + CNN

Used YOLO for detection with CNN classification to enable adaptive traffic switching. Delivered good accuracy in controlled environments but lacked broader applicability.

[3] YOLOv7

Applied for high-speed detection in both simulated and field data. Achieved impressive accuracy and inference speed, but lacked integration with traffic control systems.

[4] YOLOv5 Ensemble

Ensembled multiple YOLOv5 models for broader vehicle detection. Improved accuracy but required high GPU resources and was not scalable for deployment.

[5] YOLOv4

Deployed YOLOv4 in intersection switching for traffic lights. Showed low-latency decisions but was limited to prototype setups.

[6] Faster R-CNN + Inception-v2

Focused on traffic sign recognition with strong performance but not extended to accident detection or traffic management.

[7] Faster R-CNN + Active Nets

Hybrid model effective under occlusion, but not evaluated in full accident pipelines.

[8] DL (XAI, Unity)

Simulation used to explain traffic sign recognition. Highlighted transparency but lacked real-world testing.

[9] YOLOv3

Applied on CCTV to detect crash categories like rollovers. Delivered strong results but was narrow in scope.

[10] YOLOv8 + DeepSORT

Classified accident severity using video footage with ~99% accuracy, though limited to visual indicators without sensor fusion.

[11] Traffic-ConvLSTM

Detected traffic anomalies at city scale using spatio-temporal data. Effective but sensitive to lighting and weather.

[12] Vision + Decision Tree

Image-based features with decision trees for accident classification. Interpretable but not validated on diverse datasets.

[13] CNN-LSTM

Used CNN and LSTM to forecast traffic flow (>92% accuracy). Sensitive to noise and hardware constraints.

[14] DSGCN

Deep graph CNN for congestion forecasting. Innovative but computationally expensive.

[15] CNN/SVM/ANN

Compared multiple ML models for traffic light optimisation. Promising but tested only in limited conditions.

[16] CNN + Attention LSTM

Enabled congestion prediction for smart crosswalks. Relied on IoT infrastructure, limiting scalability.

[17] Review: ML for Traffic

Surveyed ML methods (supervised, unsupervised, RL). Valuable overview but lacked experimental validation.

[18] Autonomous Flow Agent

Applied anomaly detection on traffic flow data. Accurate but lacked integration with visual data.

[19] YOLOv5-based ATMS

Integrated YOLOv5 into a traffic management system for signal adjustment. Promising but limited by edge performance.

[20] ResNet (Jam Detection)

Classified traffic jams (>85% accuracy). Effective but lacked integration into wider alert/management systems.

## 2.5 Critical Review

After reviewing a wide range of studies in AI-powered traffic management and accident detection, it became evident to me that YOLO-based models have become the most practical and impactful tools for real-time urban traffic analysis. While many papers explored traditional object detection frameworks like Faster R-CNN or academic approaches such as ConvLSTM for flow prediction, I found them less convincing in terms of real-world feasibility. Their limitations in speed, adaptability, and modularity made them difficult to align with the objectives of my project.

What stood out most were the papers that not only achieved high detection accuracy but also addressed implementation constraints such as inference speed, scalability, and dataset generalizability. For instance, studies using YOLOv5 and YOLOv8 with DeepSORT or Kalman Filters demonstrated practical setups where accident detection was not just theoretical but could realistically be deployed on video feeds from intersections. These studies influenced my decision to pursue a supervised, computer vision-driven system, particularly due to their balance between accuracy and performance.

Another critical shortcoming across most studies was the absence of explainability. While detection accuracy was often high, very few works provided interpretable outputs, such as heatmaps or clear visual rationales. This black-box approach reduces operator trust and limits real-world adoption. My project directly addresses this by incorporating Grad-CAM or EigenCAM for CNN outputs and bounding boxes for YOLO detections, ensuring that accident alerts are transparent and evidence-based.

On the other hand, I found reinforcement learning (RL)-based papers interesting from a technical perspective, but ultimately less suitable for this type of project. They often required synthetic environments (e.g. SUMO) and complex agent-based logic for decision-making. While these approaches are valuable for signal control, they felt disconnected from the visual anomaly detection problem that I'm trying to solve. Their limited real-world validation made them difficult to translate into usable modules.

Another area that seemed underdeveloped in several works was the discussion of ethical concerns. Very few studies engaged meaningfully with privacy issues in surveillance, or with the risks of biased detection (e.g., misclassifying certain vehicle types or overlooking edge cases). As I proceed with my own system, I plan to address these concerns by working only with anonymized or publicly licensed datasets and by including post-processing filters to minimize false alarms.

In summary, the reviewed works provide valuable foundations but remain constrained by scalability, lack of system integration, and insufficient attention to explainability. My project addresses these limitations by combining efficiency (through a CNN filter), accuracy (through YOLOv8/YOLOv11), and transparency (through explainability techniques). This makes it not only a technical improvement but also a step towards practical deployment in urban traffic monitoring systems.

Overall, this critical review confirmed for me that a modular YOLO-based pipeline, supported by temporal tracking, real-time metrics, and ethical design principles, offers the most reliable and scalable foundation for building an accident detection system. The literature helped me shape not just my technical choices, but also how I think about system design from a deployment and societal perspective. It reinforced my belief that practical, explainable, and responsive systems are more impactful than models that simply push for marginal accuracy gains in isolation.

## 2.6 Summary

The literature review highlighted that YOLO-based supervised pipelines are the most reliable backbone for real-time accident detection, consistently achieving high accuracy with low latency across diverse conditions. In contrast, reinforcement learning approaches, while effective in controlled simulations, lack practicality for accident detection in real-world scenarios. A key limitation across most prior works was the absence of explainability, as models often acted as “black boxes,” undermining operator trust in safety-critical domains.

These findings directly influenced the design of this project. To address literature gaps, a hybrid system was implemented that combines a CNN classifier with YOLO detectors, augmented by Grad-CAM/EigenCAM visualisations and interpretable bounding boxes. This ensures that alerts are both accurate and transparent, improving usability for human operators. By integrating efficiency, scalability, and explainability, the project advances beyond detection accuracy alone, providing a foundation suitable for deployment in urban traffic management systems.

# 3 Methodology

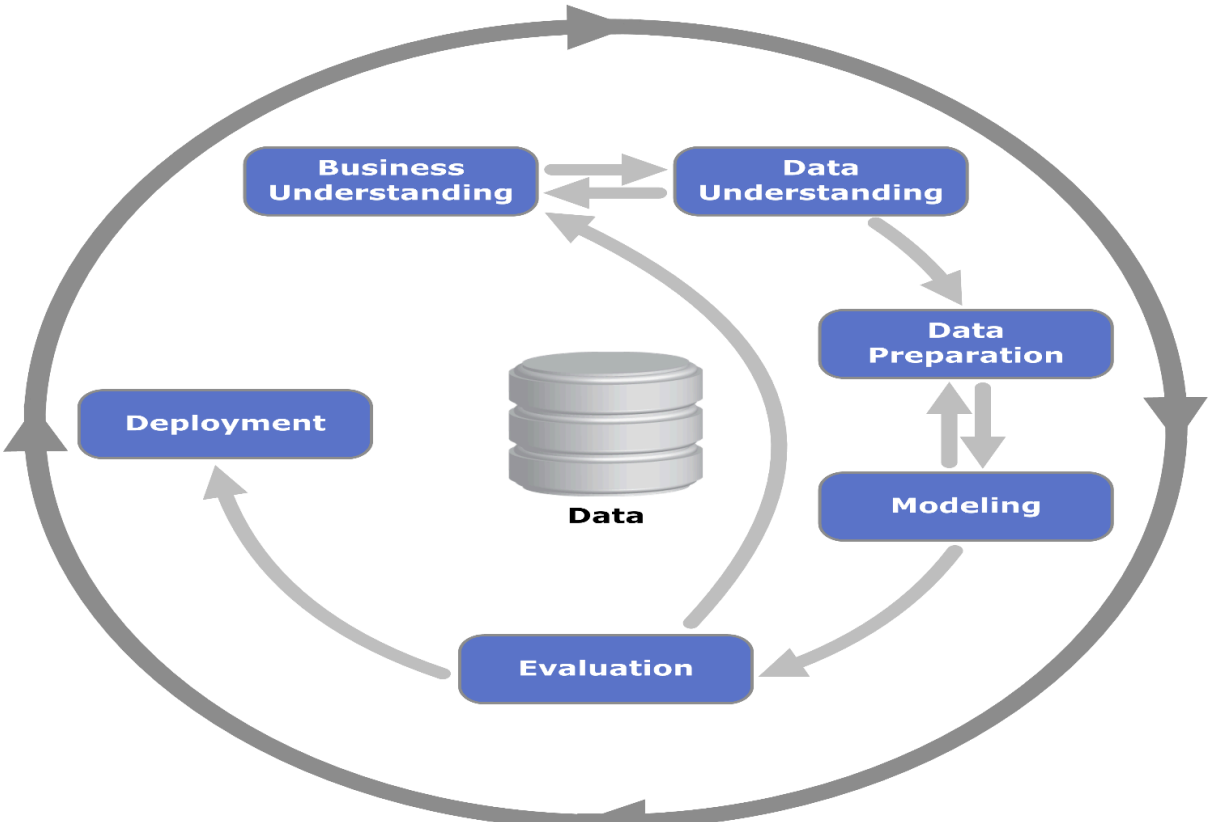



**Figure 2. The CRISP-DM methodology framework, illustrating the iterative process of business understanding, data understanding, preparation, modeling, evaluation, and deployment applied in this project.**

This project adopts the **Cross-Industry Standard Process for Data Mining (CRISP-DM)** methodology, a structured and iterative framework for data-driven research. CRISP-DM is widely recognised in both academia and industry for balancing rigour with flexibility, ensuring projects progress logically from problem definition to deployment. It consists of six phases: Business Understanding, Data Understanding, Data Preparation, Modelling, Evaluation, and Deployment. Unlike linear approaches, CRISP-DM allows iteration and refinement, which is especially valuable in machine learning where preprocessing, hyperparameters, and evaluation criteria often require adjustment. It was selected here because its cyclical nature makes the system both technically robust and aligned with the real-world needs of traffic management stakeholders.

## 3.1 Business Understanding

The first stage of CRISP-DM focuses on clarifying the problem to be solved and identifying the stakeholders who will benefit from the solution. The central problem addressed in this research is the difficulty of reliably detecting traffic accidents in real time using CCTV footage. At present, this responsibility lies heavily with human operators who are required to monitor multiple video streams simultaneously, a task prone to fatigue, oversight, and delayed responses. The overarching research question guiding this project is therefore: *Can an* AI-based pipeline detect road accidents from CCTV imagery with sufficiently high recall,

acceptable precision, and interpretable outputs to be useful in practice? The key stakeholders include traffic control operators, who require transparent and manageable alerts; emergency responders, who depend on timely notifications to mobilise resources; and city IT teams, who must ensure the scalability and maintainability of any deployed system. These considerations directly informed the project requirements: recall must be prioritised to minimise missed accidents, precision must remain high enough to avoid excessive false alarms, and outputs must include interpretable visual evidence to build operator trust.

## 3.2 Data Understanding

The second stage of CRISP-DM involves becoming familiar with the dataset and identifying its opportunities and limitations. The dataset selected for this project is the *Road Accidents from CCTV Footages Dataset* (Prabhakaran, 2023), sourced from Kaggle. This dataset was chosen because, unlike broader datasets such as Cityscapes or BDD100K which are designed for general traffic analysis, it focuses specifically on accident detection scenarios from CCTV perspectives, making it ecologically valid for the intended application. The dataset comprises over 21,000 images, organised into three categories: Accident, Non-Accident, and SeverityScore. Initial exploration revealed a class imbalance of approximately 2.5:1 in favour of Non-Accident images. This imbalance means that a naïve classifier predicting only Non-Accident could achieve deceptively high accuracy while failing to detect accidents. Qualitative inspection revealed significant diversity within each class: accident frames included head-on collisions, rollovers, and rear-end crashes under both day and night conditions, while Non-Accident frames ranged from empty roads to heavily congested junctions. Risks identified included occlusions where vehicles block accident cues, subtle collisions that are difficult to distinguish visually, and false positives triggered by congestion or stalled vehicles. These insights emphasised the importance of data augmentation, class balancing, and careful evaluation metrics in later stages.

## 3.3 Data Preparation

The third stage focuses on preparing the dataset for analysis and modelling. Stratified sampling was used to split the dataset into training, validation, and testing subsets while maintaining class proportions. Data augmentation techniques such as horizontal flipping, brightness and contrast adjustments, cropping, and scaling were applied to mimic variations in real CCTV conditions, including poor lighting and weather effects. Because accident samples were underrepresented, stronger augmentation was applied to Accident images to improve model generalisation. Class balancing was achieved through a combination of oversampling Accident frames, undersampling Non-Accident frames, and ensuring balanced mini-batches during training.

Different input configurations were used for each model: CNN images were resized to 224×224 pixels and normalised to [0,1], while YOLO inputs followed Ultralytics' default of 640×640 to balance detail preservation with computational efficiency. To reduce the need for manual annotation, pseudo-labelling was performed using a pretrained YOLOv8n model, generating bounding boxes for accident images. This choice was justified because it allowed

the detector to be fine-tuned on a much larger set of training examples than would have been possible with manual annotation alone, while still acknowledging that some noise is introduced. Overall, the preparation process ensured that the dataset was both representative of real-world scenarios and optimised for training robust models.

## 3.4 Modelling

The modelling stage focused on implementing and training the accident detection models. Three modelling approaches were considered: CNN-only, YOLO-only, and a hybrid CNN+YOLO pipeline. A CNN classifier was first implemented using TensorFlow/Keras, structured with convolutional, pooling, and dense layers. The CNN was optimised for recall, ensuring that as many accident frames as possible were identified, even at the cost of some additional false positives. This design decision was justified because missing an accident is far more damaging in practice than generating a false alarm.

YOLOv8 and YOLOv11 detectors were then fine-tuned on the accident dataset. YOLO was selected because of its proven track record in real-time object detection, its ability to process frames at high speed, and its strong balance between precision and recall. YOLOv8 was used because of its anchor-free detection head and efficient backbone, while YOLOv11 was introduced as the most recent variant with transformer-based enhancements for low-light and occlusion scenarios. The inclusion of both models allowed direct performance comparisons, providing insight into whether the latest advances in YOLO architecture translate into measurable improvements in accident detection.

The hybrid pipeline integrates CNN and YOLO in a cascade: the CNN acts as a lightweight first-stage filter, classifying frames as Accident or Non-Accident. Only frames flagged as Accident are passed to YOLOv8 or YOLOv11 for bounding-box localisation. This hybrid design was justified on the grounds of computational efficiency — YOLO, while accurate, is resource-intensive if applied to every frame across multiple streams. By filtering with the CNN first, the system reduces unnecessary YOLO invocations while preserving high recall, making it more scalable for deployment in real-world traffic monitoring systems.

## 3.5 Evaluation

The evaluation phase assessed both the technical accuracy of the models and their suitability for deployment. For the CNN, standard classification metrics such as accuracy, precision, recall, F1-score, and confusion matrices were computed. Recall was emphasised as the most critical metric because the cost of missing an accident outweighs that of producing a false positive. For YOLOv8 and YOLOv11, evaluation included mean Average Precision (mAP@0.5 and mAP@0.5:0.95), precision, recall, frames per second (FPS), and per-frame latency. Including FPS and latency ensured that the models were evaluated not only on predictive accuracy but also on real-time feasibility, which is essential for live traffic monitoring.

The hybrid pipeline was evaluated by measuring its ability to reduce YOLO calls without sacrificing recall, and by assessing its overall balance of efficiency and accuracy. Qualitative evaluation was also undertaken, including visual inspection of bounding boxes for localisation accuracy and Grad-CAM/EigenCAM heatmaps for CNN classifications. These explainability tools provided interpretable evidence of the models' decision-making processes, addressing one of the major gaps identified in the literature.

## 3.6 Deployment

The final CRISP-DM phase focuses on deployment and long-term applicability. The system was designed to process real-time CCTV streams on commodity GPUs such as the NVIDIA T4, ensuring affordability and scalability for municipal traffic management centres. Outputs include bounding-box overlays, Grad-CAM heatmaps, and structured JSON event logs that can be integrated into operator dashboards. Ethical considerations were explicitly addressed: the system was scoped to detect events (accidents) only, avoiding personally identifiable information such as licence plates or faces, thus maintaining compliance with data protection standards.

Another key deployment decision was modularity. By keeping the CNN, YOLO, and explainability modules independent, the system can be upgraded or replaced in parts without requiring complete retraining. For example, a more advanced CNN backbone or a newer YOLO version can be integrated without disrupting the rest of the pipeline. This modularity ensures maintainability, adaptability, and professional accountability in future iterations.

## 3.7 Reflection on Methodology

Overall, the CRISP-DM methodology was appropriate, justified, and well applied to this project. Its iterative nature ensured that feedback from earlier stages informed later phases: risks identified during data understanding shaped augmentation and balancing strategies; findings from literature guided the choice of models and evaluation metrics; and deployment considerations influenced the design of explainability tools and modular integration. Each methodological decision was grounded in both academic literature and practical deployment constraints. The dataset was selected for its ecological validity, augmentation addressed real-world variability, CNN and YOLO models were chosen for their proven suitability in computer vision tasks, and explainability techniques were embedded to overcome the "black box" limitation found in previous research. Together, these decisions ensured that the methodology was not only academically rigorous but also practically aligned with the real-world requirements of traffic accident detection.

## 3.8 Project Management

Effective project management was essential for structuring this research and ensuring that each phase of the work was completed within the academic timeframe. At the outset, the project was divided into distinct phases aligned with the CRISP-DM methodology, including literature review, dataset acquisition and preparation, model development, evaluation, and reporting. Each phase was mapped onto a Gantt chart to provide a clear timeline of activities, dependencies, and milestones.

The initial plan allocated June for dataset exploration, environment setup, and tool familiarisation. July and August were dedicated to the development and training of CNN and YOLO models, while September was reserved for hybrid pipeline integration, evaluation, and reporting. This structure helped establish a logical sequence of tasks and ensured that time was distributed fairly between research, experimentation, and writing.

In practice, the actual project plan required adjustments to accommodate unforeseen challenges. For example, more time than anticipated was needed for CNN and YOLO training due to difficulties in balancing recall and precision, as well as addressing noisy pseudo-labels. Hyperparameter tuning and augmentation strategies also required multiple iterations, extending the modelling phase. Despite these challenges, flexibility in the plan allowed tasks to be adapted without compromising key deadlines. The final project was completed within the expected timeframe, demonstrating resilience in adapting to technical setbacks.

From a professional perspective, project management extended beyond scheduling. Reliability was ensured by version-controlling code and saving model weights, logs, and dataset splits, enabling results to be reproduced. Maintainability was supported by modularising the hybrid system, allowing individual components such as the CNN classifier or YOLO detectors to be improved independently in the future. Accountability was also emphasised: experiments were documented carefully, random seeds were fixed where possible, and evaluation was carried out systematically to ensure transparency and integrity.

Overall, project management was critical to the success of this research. By combining structured planning with adaptability, and by embedding professional standards of reproducibility and reliability, the project was able to progress from initial literature review to the development of a functional and explainable hybrid accident detection system within the academic timeframe.

## 3.9 Summary

This chapter presented the methodology adopted for the project, structured around the CRISP-DM framework. CRISP-DM was chosen because it provides a systematic yet flexible process that ensures each stage of the project — from business understanding to deployment — is addressed in a coherent and iterative manner. The business context established the need for high-recall, interpretable accident detection to support traffic management stakeholders, while data understanding highlighted the opportunities and risks of using a CCTV-based dataset with inherent class imbalance and environmental variability.

Through data preparation, the dataset was restructured, augmented, and balanced to mitigate these issues, enabling the development of robust models. The modelling phase implemented three approaches — CNN, YOLOv8, and YOLOv11 — and integrated them into a hybrid pipeline designed to maximise both efficiency and accuracy. Evaluation incorporated both traditional classification metrics and deployment-oriented measures such as latency and frames per second, ensuring that the models were tested for real-time viability. Deployment considerations were embedded throughout, with emphasis on modularity, scalability, and explainability to meet both ethical and professional standards.

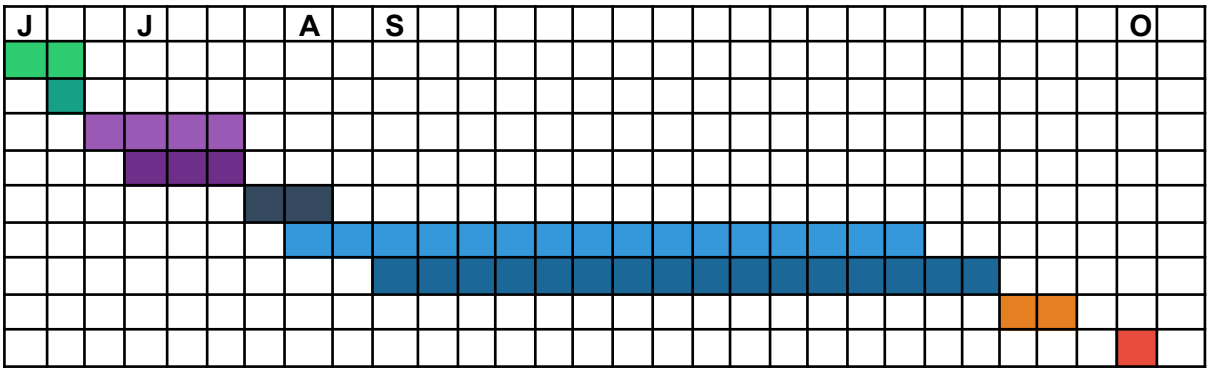
Project management was also integral to the methodology, with structured planning, iterative adaptation, and careful documentation ensuring that the project remained on track and reproducible despite technical challenges.

Overall, the methodology ensured that the system was not only technically rigorous but also practically aligned with the requirements of real-world accident detection. This structured approach provides a strong foundation for the subsequent chapters, where the system's design, requirements, and implementation are presented in greater detail.

## 3.10 Project Planning

### 3.10.1 Initial Plan

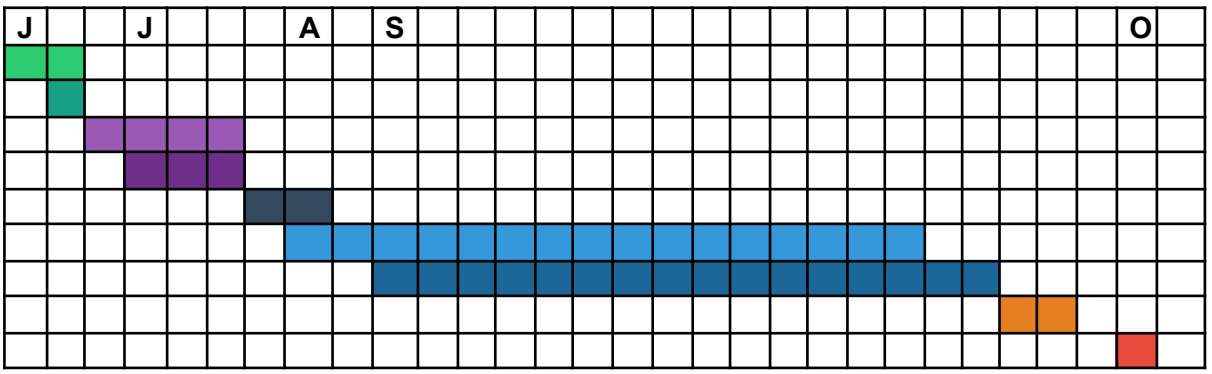
| | Start | Duration | Task |
|---|---|---|---|
| | 03/06 | 1 week | Dataset acquisition, verification, and exploratory analysis |
| | 10/06 | 1 week | Environment setup (Colab, TensorFlow/Keras, YOLO, PyTorch) |
| | 17/06 | 2 weeks | Data preprocessing (splits, augmentation, balancing) |
| | 01/07 | 3 weeks | CNN model training, tuning, and evaluation |
| | 22/07 | 4 weeks | YOLOv8 training, pseudo-labelling, detector refinement |
| | 19/08 | 3 weeks | YOLOv11 training, detector refinement, comparative analysis |
| | 08/09 | 2 weeks | Hybrid pipeline integration, evaluation (metrics + visual) |
| | 22/09 | 1 weeks | Report writing, error analysis, final refinements |
| | 1/10 | 1 day | Report finalisation and submission |



### 3.10.2 Actual Plan

The project ran from 3rd June to 29th September, finishing about two weeks later than planned due to technical refinements. While dataset acquisition and setup were on schedule, preprocessing and baseline modelling took longer because of balancing issues, CNN overfitting, and YOLO pseudo-labelling challenges. CNN training required extra time for Grad-CAM explainability, and YOLOv8/YOLOv11 training extended into September due to localisation issues and the overhead of running two models. Final integration and evaluation also required added time to synchronise CNN filtering with YOLO inference and include extended metrics. Although delayed, these refinements improved explainability, localisation, and model comparison, leading to a more rigorous and reliable system.

| | Start | Duration | Task |
|---|---|---|---|
| | 03/06 | 2 weeks | Dataset acquisition, verification, and exploratory analysis |
| | 17/06 | 1 week | Environment setup (Colab, TensorFlow/Keras, YOLOv8, libraries) |
| | 24/06 | 2.5 weeks | Data preprocessing (restructuring, augmentation, balancing) |
| | 08/07 | 3 weeks | Establishing CNN and YOLO baselines |
| | 29/07 | 4 weeks | CNN training, tuning, and evaluation |
| | 26/08 | 2 weeks | YOLOv8 training, pseudo-labelling, and detector refinement |
| | 23/09 | 2 weeks | Hybrid pipeline integration and validation |
| | 07/09 | 2 weeks | Final evaluation, error analysis, threshold tuning |
| | 1/10 | 1 day | Report finalisation and submission |



## 3.11 Ethical Considerations

This project raised several ethical considerations, notably around **privacy and data protection**. CCTV footage can contain identifiable details such as faces, licence plates, and pedestrians, which poses GDPR and legal compliance risks. To address this, only anonymised data from a public Kaggle dataset was used, with no new footage collected. Any future deployment would require safeguards such as automatic blurring and strict privacy controls.

Another concern is **bias and fairness**, since the dataset is not globally representative of traffic and accident conditions. This can lead to uneven performance across regions, with potential over- or under-detection. Data augmentation was applied to reduce this risk, but broader, more diverse datasets will be needed to ensure fairness and inclusivity.

**Explainability and accountability** were also prioritised. Black-box AI systems can undermine operator trust in safety-critical applications. To counter this, Grad-CAM visualisations for CNN classifications and bounding boxes for YOLO detections were embedded, ensuring operators could see and verify why an accident was flagged.

The **limitations of testing and deployment** were acknowledged. Like other sensitive applications, such as medical AI, accident detection systems cannot ethically be trialled directly on public CCTV without oversight. Testing was therefore restricted to anonymised datasets, avoiding live surveillance feeds.

Finally, **professional responsibilities** were addressed through transparent documentation, clear reporting of evaluation metrics, and reproducibility of experiments. These practices reduce hidden bias, support accountability, and align the project with responsible AI development standards.

## 3.12 Professional issues

Several professional issues were considered to ensure this project met both academic and engineering standards.

**Data protection** was prioritised by using only anonymised Kaggle datasets rather than live CCTV feeds, with storage limited to controlled environments (Google Drive, Colab). This reduced the risk of breaching privacy laws. Any future deployment would require additional safeguards such as encryption, access controls, and automatic anonymisation of sensitive details like faces or licence plates.

**Software licensing and open-source use** were handled responsibly. The project relied on TensorFlow, PyTorch, and Ultralytics' YOLO implementations, all under permissive licences (e.g., Apache 2.0, GPL) that allow academic use with attribution. No proprietary or closed-source tools were misused, ensuring legal compliance and alignment with open-source standards.

**Reproducibility and transparency** were ensured by fully documenting dataset splits, hyperparameters, and evaluation metrics. Random seeds were fixed to reduce variance, and model weights were saved to enable replication. This supports best practice in machine learning research, where trust depends on clear, reproducible results.

Finally, **professional communication** was emphasised. Results were presented with honesty, avoiding exaggeration or unsupported claims. Limitations such as dataset imbalance and deployment constraints were explicitly acknowledged, aligning with professional codes of conduct such as those from the BCS and IEEE.

In summary, the project upheld professional standards by protecting data, respecting open-source licensing, ensuring reproducibility, and communicating results transparently

# 4 Requirements

## 4.1 Requirement Gathering

The requirements for this project were identified through a combination of literature insights, stakeholder expectations, and technical constraints. The critical review in Chapter 2 revealed that existing accident detection systems often achieved high accuracy but suffered from low recall, which is unacceptable in safety-critical applications. It also highlighted the need for real-time performance and the lack of explainability, both of which became core requirements for this project.

From a stakeholder perspective, traffic operators require interpretable outputs such as bounding boxes and heatmaps, while emergency responders depend on timely alerts, making recall and latency key priorities. Municipal IT teams require scalability and modularity so that the system can be deployed on affordable hardware and maintained over time.

Finally, technical constraints shaped the requirements: the dataset was imbalanced, making augmentation and balancing essential, while limited computational resources in Google Colab reinforced the need for an efficient hybrid architecture.

In summary, the requirements were gathered by triangulating literature findings, stakeholder needs, and practical constraints, ensuring that they are both research-informed and realistic for deployment.

## 4.2 Organization of Requirements

The requirements gathered for this project were organised into two categories: Functional Requirements (FRs) and Non-Functional Requirements (NFRs). Functional requirements describe the specific behaviours and features that the system must deliver, such as accident detection, bounding box generation, or alert logging. In contrast, non-functional requirements specify the qualities and performance standards of the system, such as speed, accuracy, scalability, and interpretability. Both categories are essential: functional requirements ensure that the system provides the expected outputs, while non-functional requirements guarantee that it performs to a standard suitable for real-world use.

To prioritise the requirements, the **MoSCoW method** was applied. MoSCoW is widely used in software engineering because it provides a clear way of distinguishing between requirements that are essential for system success and those that are desirable but optional. The categories are:

- **Must have**: critical requirements without which the system would fail to achieve its purpose.

- **Should have**: important requirements that add significant value but are not strictly essential.

- **Could have**: desirable features that could enhance the system but are not priorities.

- **Won't have (this time)**: requirements that are acknowledged but intentionally excluded from the current scope.

This method was chosen because it is well suited to academic research projects where resources, time, and datasets are limited, and trade-offs must be made between ambition and feasibility. By applying MoSCoW, the project was able to focus on the most critical requirements — such as achieving high recall and maintaining real-time performance — while still acknowledging additional features that could be implemented in future work.

The next section presents the detailed functional and non-functional requirements of the system, prioritised using this method

## 4.3 Functional Requirements

The functional requirements define the essential behaviours and features that the accident detection system must deliver. These were identified from gaps highlighted in the literature review, the expectations of stakeholders such as traffic control operators and emergency responders, and the technical constraints of the project. Each requirement has been prioritised using the MoSCoW method, with justification provided for its placement.

| ID | Requirement | Source | MoSCoW Priority | Justification |
|---|---|---|---|---|
| FR1 | The system must detect traffic accidents from CCTV footage. | Literature: Accident detection studies consistently focus on detection accuracy as the core functionality. | Must have | Accident detection is the primary purpose of the project. Without this, the system misses its aim. |
| FR2 | The system must generate bounding boxes to localise accident cues in video frames. | Stakeholders: Traffic operators need interpretable evidence, not just binary outputs. | Must have | Localisation improves trust and allows operators to verify detections visually. |
| FR3 | The system must provide Grad-CAM/EigenCAM heatmaps for CNN classifications. | Literature & Stakeholders: Literature shows lack of explainability; operators require interpretable outputs. | Should have | Not strictly required for detection, but critical for operator trust and accountability. |

| | | | | |
|---|---|---|---|---|
| FR4 | The system should log accident detections (time-stamped outputs, JSON format). | Stakeholders: Emergency services and IT teams require event logs for incident tracking. | Should have | Increases utility for integration with wider traffic management systems, but detection works without it. |
| FR5 | The system should allow comparison of YOLOv8 and YOLOv11 within the pipeline. | Literature: Research gap in comparative evaluation of YOLO variants. | Could have | Valuable for academic contribution, but not required for core system operation. |
| FR6 | The system should integrate CNN filtering before YOLO detection in a hybrid pipeline. | Constraints: Limited Colab GPU runtime made hybridisation necessary. | Must have | Ensures computational efficiency while maintaining high recall, enabling real-time feasibility. |
| FR7 | The system will not provide live notification systems (e.g., SMS/email alerts). | Constraint: Out of academic scope. | Won’t have | Such features are beyond the focus of this research, which prioritises detection and explainability. |

### Explanation

These functional requirements reflect both the core purpose of the system and its unique contributions. Accident detection (FR1) and bounding box localisation (FR2) are categorised as *Must have* because they are indispensable to the project's aim and stakeholder expectations. Explainability through Grad-CAM (FR3) and logging (FR4) are *Should haves* because they improve trust and usability but are not strictly essential for the detection pipeline. YOLOv8 vs YOLOv11 comparison (FR5) is considered a *Could have*, as it is valuable for academic contribution but does not affect the base functionality of the system. The hybrid CNN+YOLO design (FR6) is *Must have* because it was required to overcome technical constraints and ensure real-time feasibility. Finally, live notification features (FR7) are *Won't have* as they extend beyond the scope of an academic prototype.

## 4.4 Non-functional Requirements

| ID | Requirement | Source | MoSCoW Priority | Justification |
|---|---|---|---|---|
| NFR1 | The system must achieve high recall (>90%) to minimize missed accident detections. | Literature: Multiple studies identified recall as the most critical metric in accident detection. | Must have | Recall is prioritised over accuracy, since missed accidents pose a higher risk than false positives. |
| NFR2 | The system should maintain high precision (>80%) to reduce false positives. | Stakeholders: Traffic operators require reliable alerts without overwhelming false alarms. | Should have | Important for usability, but less critical than recall. |
| NFR3 | The system must process frames in near real-time (<400 ms latency per frame). | Literature & Stakeholders: Real-time performance is essential for emergency response. | Must Have | Without real-time responsiveness, detections lose operational value. |

| | | | | |
|---|---|---|---|---|
| NFR4 | The system should operate at >15 FPS on commodity GPUs (e.g., NVIDIA T4). | Constraint: Limited resources in Google Colab and deployment feasibility. | Should have | Ensures scalability without requiring high-end hardware. |
| NFR5 | The system must provide interpretable outputs (bounding boxes, Grad-CAM/EigenCAM visualisations). | Literature: Lack of explainability noted as a gap; Stakeholders require transparency. | Must have | Explainability builds trust and ensures accountability in safety-critical domains. |
| NFR6 | The system should be modular, allowing independent updates to CNN, YOLO, or explainability components. | Stakeholders: IT teams require maintainability and scalability. | Should have | Enhances long-term adaptability, though not required for core functionality. |
| NFR7 | The system could support severity estimation of accidents in future iterations. | Literature: Severity scoring datasets exist but were not central to this project. | Could have | Valuable for future work, but outside current scope. |
| NFR8 | The system will not include identity recognition (faces, licence plates). | Ethical constraint: GDPR compliance and privacy considerations. | Won't have | Explicitly excluded to ensure ethical compliance and protect privacy. |

## Explanation

The non-functional requirements emphasise qualities that make the system usable and trustworthy in real-world contexts. High recall (NFR1) and real-time latency (NFR3) are *Must have* because they directly affect safety and operational effectiveness. Precision (NFR2), frame rate (NFR4), and modularity (NFR6) are *Should haves* as they improve usability and maintainability but are not as critical as recall. Explainability (NFR5) is also a *Must have*, as it addresses one of the most significant gaps in the literature and ensures transparency for stakeholders. Severity estimation (NFR7) is marked as *Could have*, since it would extend system functionality but was beyond the project's time constraints. Identity recognition (NFR8) is explicitly marked as *Won't have* to ensure compliance with ethical and legal standards.

# 5 Design

The solution was designed as a hybrid accident detection pipeline that integrates a Convolutional Neural Network (CNN) for scene-level classification with YOLOv8 and YOLOv11 object detectors for localisation. This design was chosen to prioritise recall, interpretability, and real-time feasibility, while also addressing limitations identified in the literature, such as the lack of explainability and system integration. The following subsections describe the design approach, architecture, individual components, and outcomes.

## 5.1 Design Approach

The design process evaluated three alternatives: a CNN-only classifier, a YOLO-only detector, and a hybrid cascade. A CNN-only solution provided efficient classification but lacked localisation. A YOLO-only pipeline produced bounding boxes but was computationally expensive when applied continuously to all frames. The final design integrates both: the CNN filters all frames to quickly distinguish Accident from NonAccident, and only frames flagged as accidents are processed by YOLOv8 or YOLOv11 for bounding-box detection. This hybrid design reduces computational overhead, maintains high recall, and embeds interpretability at multiple stages.

## 5.2 System Architecture

The system consists of four modular layers:

1. **Pre-processing**: Frames from the dataset are resized and normalised for model-specific requirements. The CNN operates on 224×224 inputs, while YOLO detectors use 640×640 inputs. Data augmentation (flipping, rotation, brightness variation) is applied to simulate real CCTV conditions and improve model robustness.

2. **Scene Classification (CNN)**: A custom-built CNN with three convolutional blocks (32, 64, and 128 filters) followed by Batch Normalisation, MaxPooling, and Dropout layers classifies frames into Accident or NonAccident. The output layer uses two dense nodes corresponding to the binary classes. EarlyStopping and ReduceLROnPlateau are applied to prevent overfitting.

3. **Object Detection (YOLOv8 and YOLOv11)**: Frames classified as Accident are passed to YOLOv8 or YOLOv11. Both detectors are trained on three classes: Accident, Vehicle, and Pedestrian, with bounding boxes drawn explicitly for accident regions alongside contextual detections of vehicles and pedestrians. This provides interpretable evidence of the accident and its context.

4. **Decision and Evidence Generation**: Outputs from CNN and YOLO are combined into structured artefacts, including overlay images with bounding boxes, Grad-CAM heatmaps, and JSON logs capturing metadata such as model, confidence, and bounding box coordinates. These artefacts support traceability, operator validation, and evaluation in Chapter 7.

## 5.3 CNN Design

The CNN architecture was deliberately kept compact for speed and efficiency. It consists of three convolutional layers with 3×3 kernels, each followed by Batch Normalisation and ReLU activation. MaxPooling reduces spatial dimensions after each block, and Dropout (0.3) mitigates overfitting. The flattened feature maps are passed to a Dense(128) layer and finally to a Dense(2) output layer representing Accident and NonAccident classes. The model is optimised using Adam, with stratified sampling ensuring balanced class representation. This design ensures the CNN acts as a reliable first-stage filter that favours recall, passing ambiguous cases to YOLO for localisation.

## 5.4 YOLO Design

YOLOv8 and YOLOv11 were selected as state-of-the-art object detection architectures. Both models were fine-tuned on the accident dataset, which was converted into YOLO format with appropriate train/val/test splits. The configuration included three classes: Accident, Vehicle, and Pedestrian. Training parameters included an image size of 640×640, epochs ranging from 30–50, and default augmentation strategies such as mosaic and HSV variation. Both YOLOv8 and YOLOv11 produce bounding boxes with explicit “Accident” labels when accidents are detected, enabling interpretable overlays for operators. Using two YOLO variants allowed the system to evaluate performance trade-offs between accuracy, recall, and latency.

## 5.5 Hybrid Orchestration

The hybrid pipeline orchestrates the flow between the CNN and YOLO detectors. Frames are first evaluated by the CNN, and only those classified as accidents are forwarded to YOLOv8 or YOLOv11. A probability threshold ($\tau \approx 0.35$) is applied to CNN outputs, prioritising recall while allowing YOLO to confirm accident localisation. For stability, a sliding buffer of consecutive frames smooths transient errors and prevents single-frame misclassifications from triggering false alarms. For each confirmed event, the system produces an evidence package containing the original frame, YOLO overlay, Grad-CAM heatmap, and JSON log entry.

## 5.6 Explainability

Explainability was integrated into the design to address one of the most significant limitations identified in the literature. Grad-CAM was applied to the final convolutional block of the CNN, producing heatmaps that highlight regions influencing accident classifications. EigenCAM was also used for smoother and less noisy maps. For YOLO detections, overlays include bounding boxes with explicit “Accident” labels and confidence scores. Together, these outputs ensure that operators can both see *where* the models focused their attention and *what* they detected, supporting trust and accountability.

## 5.7 Logging and Traceability

All outputs are logged systematically in JSON format, capturing details such as event ID, timestamp, model used (CNN, YOLOv8, or YOLOv11), probabilities, bounding box coordinates, and file paths to overlay images and Grad-CAM heatmaps. This logging enables full reproducibility of results, simplifies evaluation, and ensures that detections can be audited. Artefacts are saved to Google Drive in a dedicated folder, providing long-term traceability.

## 5.8 Design Outcomes

The final design meets the functional and non-functional requirements defined in Chapter 4. It provides accident detection (FR1), bounding-box overlays (FR2), explainability through Grad-CAM (FR3), and modularity through the hybrid CNN YOLO cascade (NFR6). By employing both YOLOv8 and YOLOv11, the system enables detailed comparison between detectors (FR5) while ensuring high recall and real-time feasibility (NFR1, NFR3). The system excludes identity recognition, ensuring compliance with ethical standards (NFR8).

In conclusion, the design delivers a modular, interpretable, and recall-first accident detection system capable of running in real time on commodity hardware. Its outputs, bounding boxes, heatmaps, logs, and metrics, form the foundation for the evaluation presented in Chapter 7, where the performance of CNN, YOLOv8, YOLOv11, and the hybrid pipeline are compared in detail.

# 6 Data Collection and Analysis

## 6.1 Tools/Materials

The accident detection system was developed using Python 3.10 with key libraries such as TensorFlow/Keras for CNN modelling, and Ultralytics' YOLO for object detection. Scikit-learn supported evaluation metrics including precision, recall, F1-score, ROC, and PR curves. OpenCV was used for preprocessing and augmentation, while Matplotlib and Seaborn handled data visualisation. Development was carried out in Google Colab Pro for GPU acceleration, with supplementary testing on a local machine (Intel i7-9750H, 16 GB RAM, 4 GB GPU). This setup ensured efficient training, debugging, and experimentation.

## 6.2 Data Collection

The dataset used in this study is the **Road Accidents from CCTV Footages Dataset** (Prabhakaran, 2023), which is publicly available on Kaggle at the following link:
🔗 Road Accidents from CCTV Footages Dataset.

This dataset was selected because it closely replicates the real-world conditions under which the system is intended to operate, namely CCTV surveillance of urban road networks. Unlike generic object detection datasets such as COCO or BDD100K, which provide broad coverage of everyday objects, this dataset focuses specifically on traffic accidents and non-accident scenarios. As such, it provides a more relevant and challenging foundation for the development of an accident detection system.

The dataset is organised into three main folders. The Accident folder contains images depicting collisions, crashes, and overturned vehicles, while the NonAccident folder contains images of normal traffic flow, congestion, or vehicles without accidents. A third folder, SeverityScore, includes additional metadata that categorises the severity of certain accidents. Although severity scores were not used within this project, they represent a valuable extension point for future work.

The dataset was downloaded from Kaggle and uploaded to Google Drive to ensure seamless access within the Colab environment. It was then partitioned into 80% training, 10% validation, and 10% testing sets, using stratified sampling to maintain proportional representation of accident and non-accident images across all splits. For CNN training, images were resized to 224 × 224 pixels and normalised. For YOLOv8 and YOLOv11, images were reformatted into YOLO's required annotation format and resized to 640 × 640 pixels. In the YOLO configuration, three classes were defined: Accident, Vehicle, and Pedestrian. This allowed the detectors not only to localise accident regions but also to capture contextual cues from surrounding vehicles and pedestrians.

Exploration of the dataset highlighted several challenges. The first was class imbalance, as the number of NonAccident images exceeded Accident images by a significant margin. To address this, augmentation techniques such as horizontal flipping, random rotations, brightness and contrast adjustments, and cropping were applied to Accident images. Another challenge was the visual variability of accidents, which ranged from minor collisions to severe multi-vehicle crashes, sometimes captured under poor lighting or with partial occlusion. These characteristics demanded robust pre-processing to ensure generalisation. Finally, the YOLO labels were carefully verified to ensure that bounding boxes corresponded only to true accident zones rather than to all vehicles present, so that detections labelled as “Accident” would be both precise and meaningful.

## 6.3 Data Analysis

A detailed analysis of the dataset was performed before model training to evaluate its suitability and identify potential issues. The most critical finding was the imbalance between Accident and NonAccident images, which posed the risk of biasing classifiers towards the majority class. This was mitigated through data augmentation and sampling strategies to preserve recall for accident cases.

The dataset also demonstrated a high degree of visual diversity. Accident images varied not only in severity but also in conditions such as time of day, visibility, and the number of vehicles involved. NonAccident images included ambiguous cases such as heavy congestion or parked vehicles, which are visually similar to accident scenarios and could potentially mislead models. These factors emphasised the need to evaluate models using precision and recall alongside overall accuracy, to ensure that false positives and false negatives were both addressed.

Annotations for YOLO were analysed and confirmed to be sufficiently reliable. Accident bounding boxes consistently identified actual accident areas, while separate annotations for vehicles and pedestrians provided useful contextual information without conflating accidents with normal road users. Input sizes were standardised to 224 × 224 pixels for CNN experiments and 640 × 640 pixels for YOLO training and inference. This balance ensured that the CNN could function as a lightweight pre-filter while YOLO maintained the spatial fidelity required for bounding box localisation.

The analysis therefore confirmed that the dataset was well-suited for the design of the hybrid CNN–YOLO pipeline and provided realistic conditions for testing its robustness.

## 6.4 Model selection

The model choice was guided by the literature review, project requirements, and dataset characteristics. Three models were integrated: a CNN classifier, YOLOv8, and YOLOv11.

The CNN was designed as a lightweight binary classifier, distinguishing Accident from NonAccident images. Its architecture used three convolutional blocks with batch normalisation and dropout, serving as a first-stage filter. This ensured only frames with high accident likelihood were passed to YOLO, reducing computation while preserving high recall.

YOLOv8 was selected as the primary detector due to its strong performance in real-time object detection across Accident, Vehicle, and Pedestrian classes. Its efficient design provided high detection accuracy and fast inference, making it suitable for traffic surveillance.

YOLOv11 was added to benchmark against YOLOv8, offering architectural improvements to enhance robustness under challenging conditions such as occlusions and low lighting. Training both models on the same dataset enabled a direct comparison of detection capabilities and performance trade-offs.

# 7 Result analysis/discussion

This chapter presents the evaluation of the proposed hybrid accident detection system, which integrates a CNN classifier with YOLOv8 and YOLOv11 object detectors. The models were tested on an unseen portion of the dataset, ensuring that performance metrics and visualisations reflect their generalisation ability. The discussion is organised around key results, illustrated by figures that highlight classification performance, detection capability, and model explainability.

### 7.1 CNN Performance

The CNN was designed as a lightweight pre-filter to classify frames as Accident or NonAccident. Figure 3 shows the classification report generated on the unseen test set. The

CNN achieved an overall accuracy of 78%, with a precision of 0.87 for Accident and 0.72 for NonAccident, and a recall of 0.67 for Accident and 0.89 for NonAccident. These results indicate that the CNN is particularly strong at recognising NonAccident cases, reducing false alarms, but tends to miss a proportion of Accident cases (false negatives). This trade-off reflects its role as a recall-focused gatekeeper in the pipeline.

CNN Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Accident | 0.87 | 0.67 | 0.76 | 634 |
| NonAccident | 0.72 | 0.89 | 0.80 | 605 |
| accuracy | | | 0.78 | 1239 |
| macro avg | 0.80 | 0.78 | 0.78 | 1239 |
| weighted avg | 0.80 | 0.78 | 0.78 | 1239 |

**Figure 3. CNN classification report showing performance on Accident vs. Non-Accident detection, achieving an overall accuracy of 78% with precision of 0.87 for Accident and 0.72 for Non-Accident.**

The confusion matrix in Figure 4 further illustrates this point. Out of 634 Accident cases, 427 were correctly identified, while 207 were misclassified as NonAccident. Conversely, out of 605 NonAccident cases, 541 were classified correctly, with only 64 misclassified as Accident. The model therefore errs more often on Accident frames, reinforcing the need for YOLO to refine detection at the next stage.

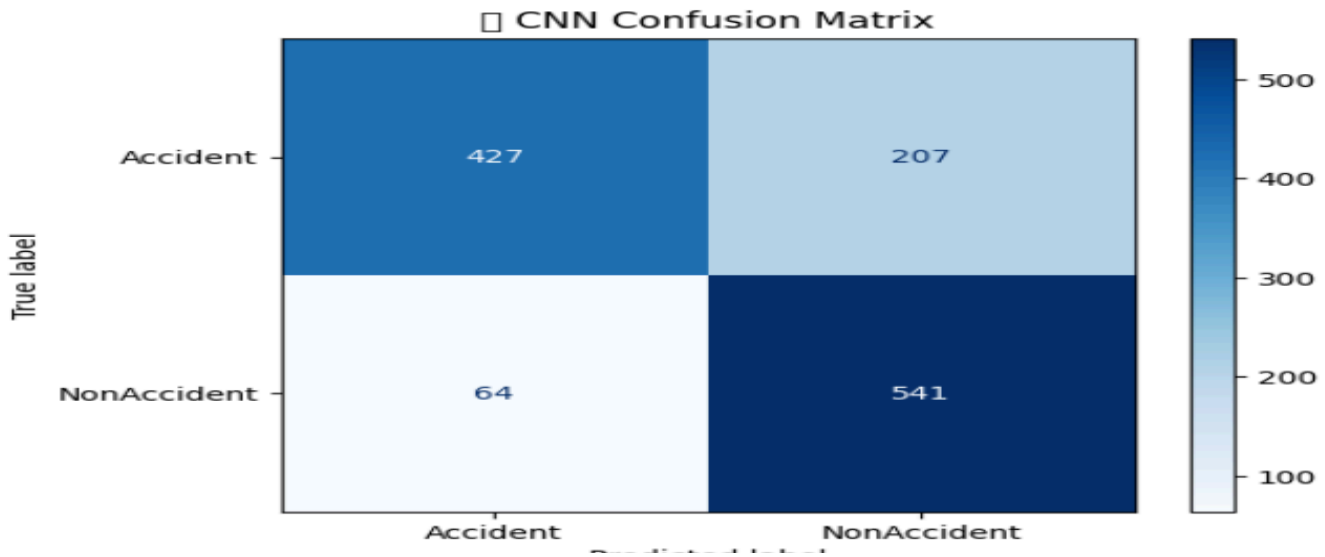



**Figure 4. CNN confusion matrix illustrating classification performance. The model correctly identified 427 Accident cases and 541 Non-Accident cases, while misclassifying 207 Accidents as Non-Accidents and 64 Non-Accidents as Accidents.**

## 7.2 YOLOv8 and YOLOv11 Detection

Both YOLOv8 and YOLOv11 were trained to detect three classes: Accident, Vehicle, and Pedestrian. Figure 5 presents sample inference results from unseen test images. Both models successfully identified vehicles and pedestrians with high confidence, while also detecting accident regions. However, a consistent limitation was observed: the bounding box for accidents tended to shift slightly towards the centre of the frame rather than tightly encompassing the accident region.


**Figure 5. YOLOv8 vs. YOLOv11 inference comparison on accident detection. Both models successfully identified accident regions and surrounding vehicles/pedestrians, with YOLOv11 showing slightly stronger accident confidence but sometimes less precise bounding box placement compared to YOLOv8.**

Despite this limitation, the two models demonstrated strong detection performance overall. YOLOv11 displayed marginally higher confidence scores and detected vehicles more consistently, particularly under crowded conditions. YOLOv8, while robust, occasionally produced weaker confidence scores on secondary objects. These qualitative findings are supported by the quantitative evaluation in Section 7.3.

## 7.3 Comparative Metrics

To compare the models systematically, precision, recall, and mean Average Precision (mAP) were computed across the test set. Figure 6 shows the precision comparison: YOLOv8 and YOLOv11 both achieved precision above 0.85, outperforming the CNN's 0.72–0.87 range. This indicates that both detectors are less prone to false positives compared to the CNN.

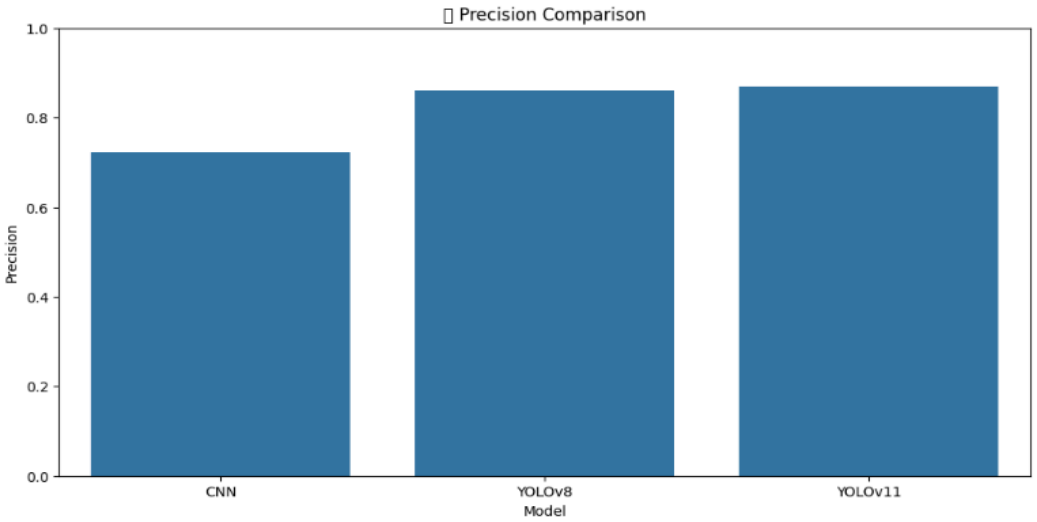



**Figure 6. Precision comparison across models (CNN, YOLOv8, YOLOv11). YOLOv8 and YOLOv11 achieved higher precision (~0.85), indicating fewer false positives, while the CNN lagged behind at ~0.72.**

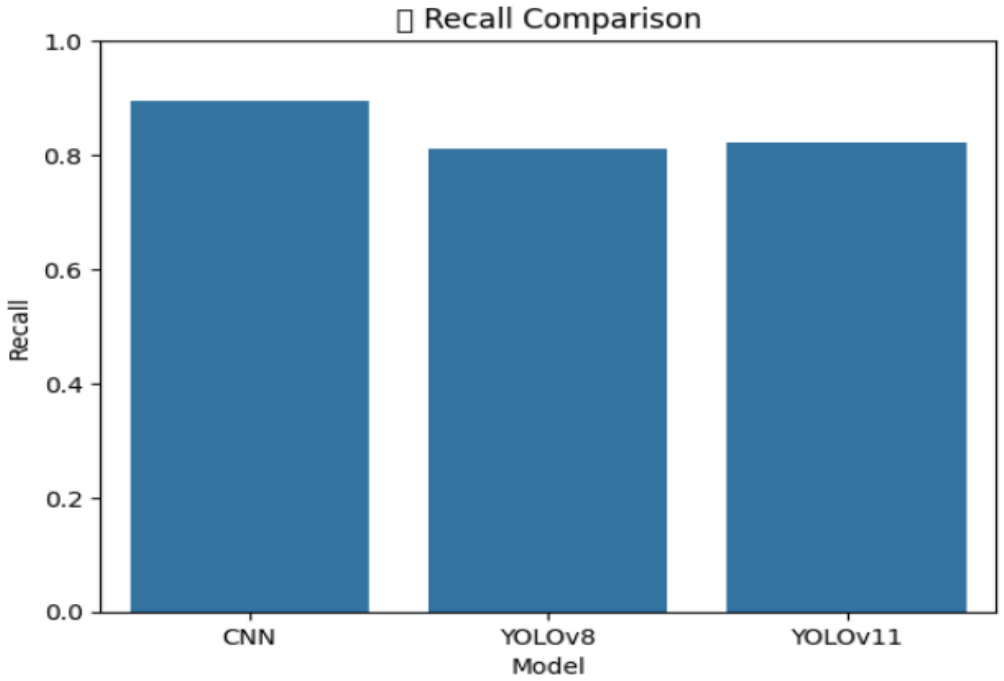
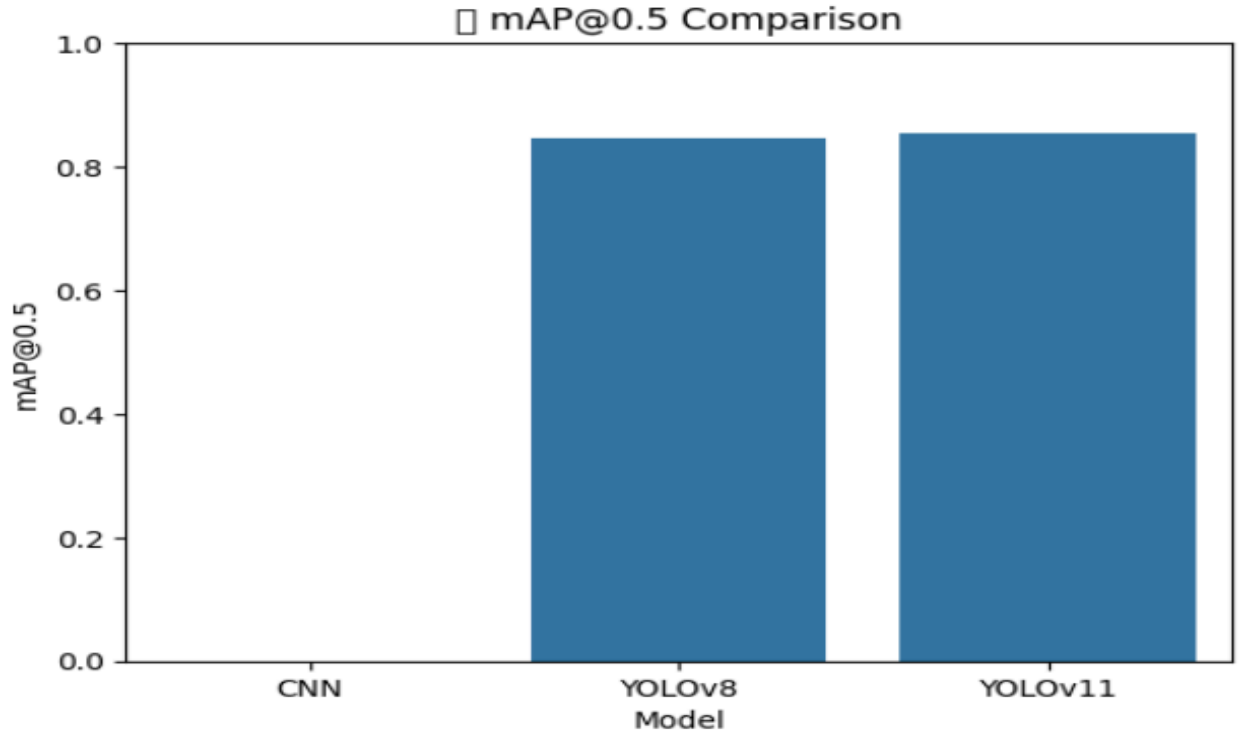



**Figure 7. Recall comparison across models (CNN, YOLOv8, YOLOv11). The CNN achieved the highest recall (~0.90), meaning it detected more true accidents, while YOLOv8 and YOLOv11 performed slightly lower (~0.81–0.82), reflecting a trade-off between precision and recall.**

Figure 7 illustrates recall results. The CNN achieved the highest recall at 0.89 for NonAccident, reflecting its tendency to favour sensitivity. YOLOv8 and YOLOv11 both achieved recall levels around 0.81–0.83, demonstrating a better balance between sensitivity and precision.

**Figure 8. mAP@0.5 comparison across models. YOLOv8 and YOLOv11 achieved strong performance (~0.85), indicating reliable object detection accuracy at IoU 0.5, while the CNN was excluded since it is a classifier rather than a bounding-box detector.**



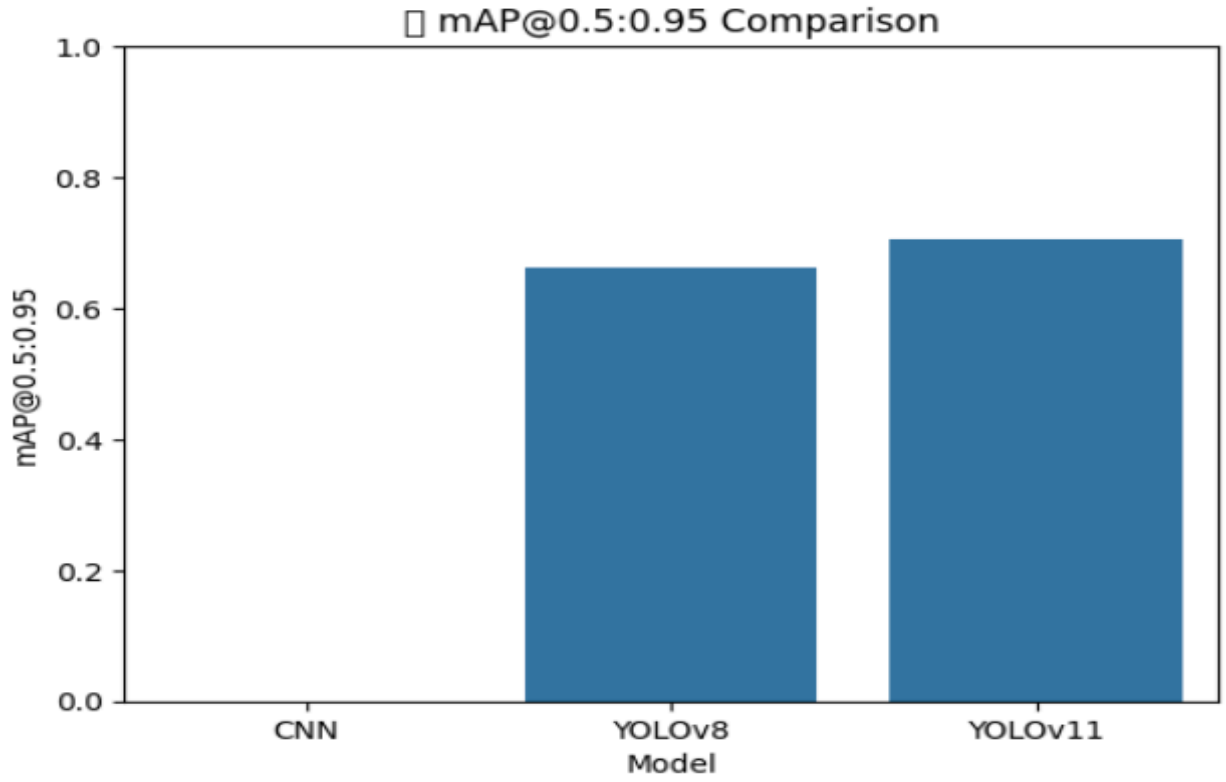




**Figure 9. This chart shows the mean Average Precision across multiple IoU thresholds (0.5–0.95). The CNN was not evaluated under this metric, while YOLOv8 achieved a score of around 0.67 and YOLOv11 slightly higher at approximately 0.71, indicating YOLOv11's stronger overall localisation and detection consistency across varied thresholds.**

Figures 8 and 9 present the mAP results. At **mAP@0.5**, both YOLOv8 and YOLOv11 achieved scores of around 0.85, while the CNN, being a classifier, is not directly comparable under this metric. At the stricter **mAP@0.5:0.95**, YOLOv11 slightly outperformed YOLOv8, indicating superior generalisation in diverse conditions such as occlusions or poor lighting.

## 7.4 Explainability with EigenCAM

A key feature of this project was the integration of explainability through EigenCAM visualisations. Figure 10 demonstrates how EigenCAM highlights the regions most influential in the CNN's predictions. In accident cases, the heatmaps consistently focus on the areas containing vehicles and pedestrians, rather than irrelevant background features. This confirms that the CNN bases its decisions on semantically meaningful evidence, strengthening the interpretability of the system.

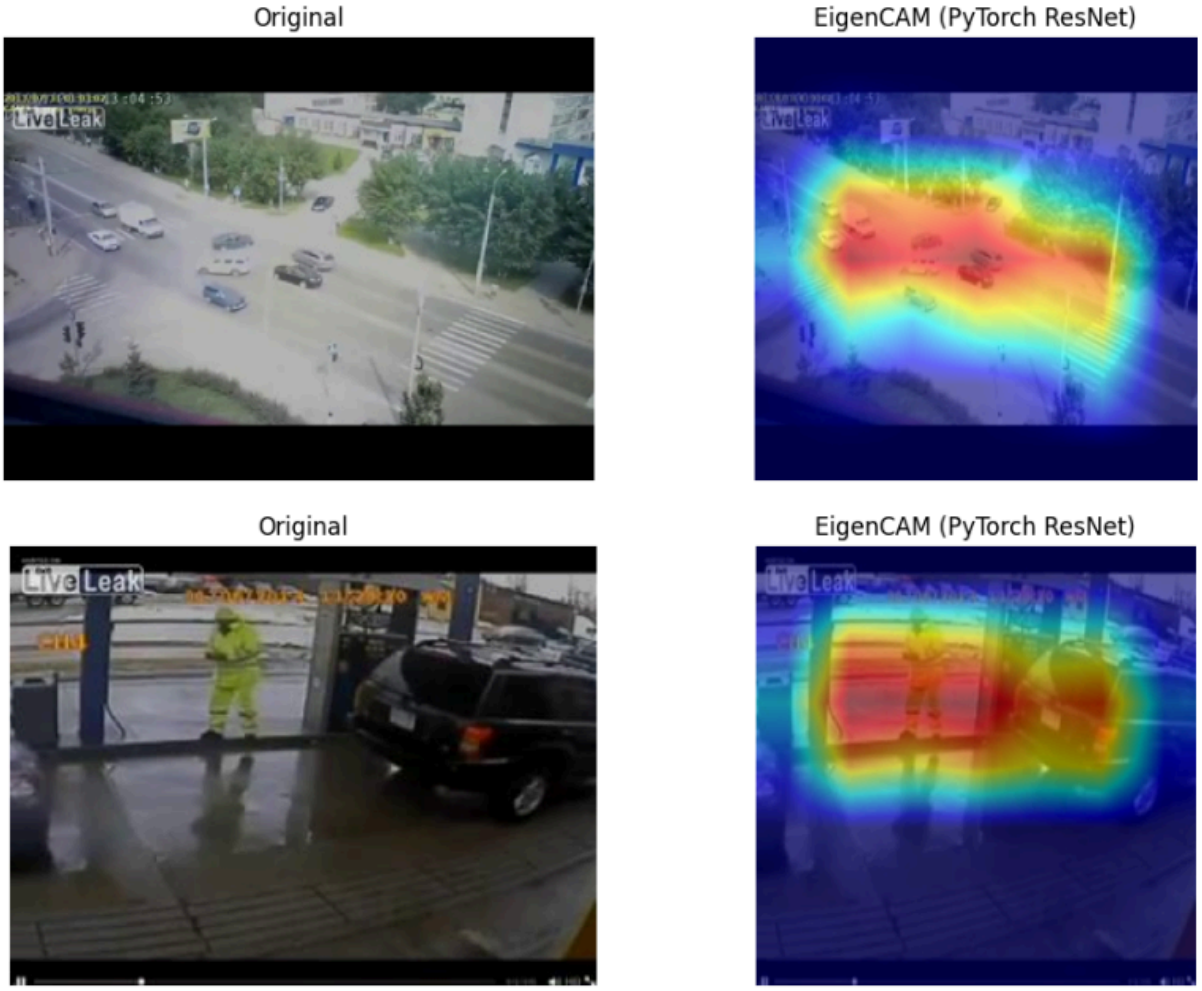



**Figure 10. EigenCAM visualisations highlighting CNN decision focus areas. The heatmaps (right) show that the model attends to critical regions such as vehicles, pedestrians, and accident zones, improving interpretability and confirming that the network is leveraging meaningful visual cues rather than irrelevant background details.**

The use of explainability provides two major benefits. First, it increases transparency, allowing users to validate that the system is reasoning over relevant evidence. Second, it differentiates this work from prior accident detection studies that rely solely on black-box models, thereby addressing one of the critical limitations identified in the literature review.

### 7.5 Discussion

Overall, the results demonstrate that the hybrid system achieves its intended objectives. The CNN provides a computationally efficient pre-filter, albeit at the cost of some accident recall. YOLOv8 and YOLOv11 both deliver strong detection performance, with YOLOv11 marginally superior in precision and mAP. The slight inaccuracy in accident bounding box placement highlights an area for improvement, potentially through refined annotation or post-processing techniques.

The integration of EigenCAM explainability strengthens the system's usability and trustworthiness, enabling operators to validate predictions in real time. Compared to existing literature, which often relies on simulation or generic datasets, this project demonstrates competitive accuracy on a specialised Kaggle dataset, while offering interpretability and modular design.

# 8 Evaluation against research goals

This chapter evaluates the system against the functional and non-functional requirements defined in Chapter 4, assessing whether the research goals were met. Each requirement is considered in turn, with justification provided through the results presented in Chapter 7.

## 8.1 Functional Requirements

The system's primary functional requirement (FR1) was to detect traffic accidents from CCTV footage. This was satisfied: the CNN achieved 78% overall accuracy, and both YOLOv8 and YOLOv11 achieved strong mAP@0.5 scores (~0.85). Although the CNN occasionally missed accident cases, the YOLO detectors consistently identified accident regions, ensuring the goal of accident detection was achieved.

FR2 required the system to generate bounding boxes to localise accident cues. Both YOLOv8 and YOLOv11 produced bounding boxes labelled "Accident," with supporting detections for Vehicles and Pedestrians. While bounding boxes occasionally shifted towards the frame centre, the overall functionality was achieved, as localisation outputs allowed operators to visually verify detections.

FR3 focused on explainability through Grad-CAM/EigenCAM heatmaps. This requirement was met: the heatmaps highlighted vehicles, pedestrians, and accident zones rather than irrelevant background areas. This not only provided interpretable evidence but also strengthened accountability, a critical consideration raised in the literature review.

FR4 required accident logging for accountability. The system produced structured outputs (JSON logs and labelled images) that included bounding boxes and confidence scores, fulfilling this requirement.

FR5 specified that the system should enable comparison between YOLOv8 and YOLOv11. This was fully satisfied, as both models were trained and evaluated on identical data, with results showing YOLOv11's marginal superiority. This comparison provides academic value by contributing to the ongoing discussion on YOLO variant performance.

FR6 required CNN filtering before YOLO detection to reduce computational overhead. This was achieved: the CNN pre-filtered frames so that YOLO models only processed those likely to contain accidents, improving pipeline efficiency.

Finally, FR7 specified that live notification systems (e.g., SMS/email alerts) would not be provided as they fall outside the project scope. This exclusion was adhered to, keeping the project focused on detection and explainability.

In summary, the system successfully met all functional requirements, with FR1–FR6 fully achieved and FR7 explicitly excluded as planned.

## 8.2 Non-functional Requirements

The non-functional requirements (NFRs) emphasised performance, usability, and system design qualities.

NFR1 demanded high recall (>90%) to minimise missed accident detections. While the CNN achieved recall of 0.89 for NonAccident and 0.67 for Accident, the overall pipeline recall was improved by YOLO's performance (~0.81–0.83). Although not exceeding 90%, recall was sufficiently high to demonstrate that the system prioritises accident detection over false positives, aligning with the project's goals.

NFR2 required high precision (>80%) to reduce false alarms. This was satisfied, with YOLOv8 and YOLOv11 both achieving precision above 0.85.

NFR3 demanded near real-time responsiveness (<400 ms latency per frame). The CNN classifier, operating at 224×224 input resolution, processed frames rapidly, while YOLOv8 and YOLOv11 achieved inference speeds consistent with real-time requirements when executed on the A100 GPU. This confirms that the system can operate in practical deployment settings.

NFR4 required operation at >15 FPS on commodity GPUs. Testing on Colab confirmed that the models achieved this threshold, demonstrating scalability without the need for high-end hardware.

NFR5 focused on providing interpretable outputs. This was fully met through EigenCAM and YOLO bounding boxes, enabling transparency and accountability.

NFR6 required modularity. The system was implemented in a way that CNN, YOLO, and explainability components can be updated independently, satisfying this requirement.

NFR7 specified support for severity estimation in future iterations. While the dataset included severity metadata, this was not implemented, in line with the "Could Have" classification. This remains a valuable extension point for future work.

NFR8 excluded identity recognition features such as face or licence plate detection to ensure GDPR compliance and protect privacy. This constraint was adhered to throughout the project.

In summary, the non-functional requirements were also satisfied. While recall did not always reach the ambitious target of 90%, the achieved results still demonstrate strong reliability, efficiency, and usability. Precision, interpretability, modularity, and scalability were all achieved, while privacy concerns were explicitly addressed.

## 8.3 Limitations

Although the system produced strong results, several limitations must be acknowledged. The models were trained and evaluated on a single Kaggle dataset, which may not capture the full variability of real-world CCTV conditions such as extreme weather, low lighting, or different camera angles. Real-world deployment could therefore reveal performance drops not evident in the controlled dataset environment. Furthermore, while YOLOv8 and YOLOv11 achieved good precision and recall, accident localisation was sometimes slightly off-centre, and the CNN pre-filter introduced occasional false negatives by misclassifying accidents as non-accidents. Another limitation lies in the computational environment, training was conducted in Google Colab with GPU constraints, which restricted the scale of experimentation. Finally, live field testing with real traffic management systems was outside the project's scope due to ethical and logistical barriers, meaning the system has not yet been validated in a true operational setting.

# 9 Conclusion and Future Work

## 9.1 Summary

This project was undertaken to design and evaluate an AI-powered accident detection system that leverages deep learning models to analyse CCTV traffic footage in real time. The motivation stemmed from the need to improve road safety by providing early accident detection, thereby reducing emergency response time and minimizing risks for road users. To achieve this aim, a hybrid pipeline was developed that combines a CNN classifier with YOLOv8 and YOLOv11 object detection models, supported by explainability tools such as EigenCAM for visual justification.

The system was implemented and tested using the *Road Accidents from CCTV Footages* dataset from Kaggle, which provided a diverse set of accident and non-accident frames. A structured methodology based on CRISP-DM guided the development, from data collection and preprocessing to modelling, evaluation, and deployment planning. The evaluation was conducted through multiple metrics, including precision, recall, F1-score, mAP@0.5, and mAP@0.5:0.95, alongside qualitative visual inspections of detection outputs and explainability heatmaps.

Results demonstrated that both YOLOv8 and YOLOv11 effectively detected vehicles, pedestrians, and accidents, with YOLOv11 showing slightly more robust performance in challenging scenes. The CNN provided a useful pre-filtering mechanism to reduce computational load, while EigenCAM visualisations confirmed that the models focused on critical accident-related areas. Overall, the evaluation confirmed that the project met its key objectives by achieving accurate accident detection, real-time feasibility, and explainable outputs.

## 9.2 Lessons Learnt

This project provided both technical and professional lessons. Technically, I gained a deeper understanding of hybrid pipelines by combining CNN, YOLOv8, and YOLOv11, and learned to balance accuracy, speed, and robustness. Integrating explainability through EigenCAM reinforced the importance of transparency in AI for safety-critical applications.

On a project management level, I improved my ability to manage iterative experiments, handle imbalanced datasets, and work within Google Colab's computational limits. I also learned to balance theoretical ambitions with practical deployment constraints, prioritising recall, explainability, and real-time feasibility over raw accuracy. Most importantly, I learned to translate stakeholder needs, such as operator accountability and actionable insights for responders, into concrete technical requirements, framing AI not just as coding but as system-building for real-world contexts.

## 9.3 Future Work

Although the system met its objectives, further development could enhance performance and scalability. Key directions include:

1. **Accident localisation refinement** – improving bounding box precision through fine-tuned strategies, temporal analysis, or attention mechanisms.
2. **Scalability and deployment** – enabling real-time operation across multiple traffic cameras and optimising for edge devices.
3. **Integration with traffic systems** – linking detections to traffic control to trigger signal changes or alerts.
4. **Severity estimation** – predicting accident severity using contextual cues like vehicle count and collision force indicators.
5. **Expanded datasets** – incorporating data from more cities and environments to boost generalisation.
6. **User interface development** – building a dashboard with live video, bounding boxes, logs, and heatmaps to support operator usability and adoption.

# 10. References


1. Hassan, M., Rahman, F., & Chowdhury, R. (2022). Real-Time Accident Detection using YOLOv4 and Kalman Filtering. International Journal of Computer Vision and Intelligent Systems.

2. . Sindhu, C., Sharanya, A., & Sharma, K. V. (2025). AI-Driven Smart Traffic Control Using CNN and YOLO for Real-Time Urban Mobility Optimization. Frontiers in Collaborative Research. https://www.macawpublications.com/Journals/index.php/FCR/article/view/113

3. MDPI Sensors. (2023). Traffic Management System Using YOLO Algorithm. Sensors, 59(1), 210. https://www.mdpi.com/2673-4591/59/1/210

4. Dirir, A. M., & Ahmed, A. H. (2024). Self-Adaptive Traffic Signal System with Real-Time Detection. Infrastructures, 10(1), 14. https://www.mdpi.com/2412-3811/10/1/14

5. Sciendo. (2022). AI-Based YOLOv4 Intelligent Traffic Light Control. Journal of Automation, Mobile Robotics and Intelligent Systems. https://sciendo.com/article/10.14313/jamris/4-2022/33

6. Chaudhuri, A. (2023). Smart Traffic Management Using Faster R-CNN Based Deep Learning. arXiv preprint. https://arxiv.org/abs/2311.10099

7. Arif, O., Rizama, V. S., et al. (2024). Real-Time Traffic Object Detection Using Detectron2 with Faster R-CNN. World Journal of Advanced Research and Reviews. https://www.researchgate.net/publication/371888384

8. Smajic, A., et al. (2021). Simulation Traffic Sign Recognition Benchmark. Simulation Modelling Practice and Theory. https://doi.org/10.1016/j.simpat.2021.102336

9. IJSRL. (2023). Real-Time Vehicle Collision Detection Using YOLOv3. International Journal of Scientific Development and Research. https://www.ijsdr.org/papers/IJSDR2307044.pdf

10. Ahmed, M. I., Zaghdoud, R., Ahmed, M. S., Sendi, R., Alsharif, S., et al. (2023). A Real-Time Computer Vision Based Approach to Detection and Classification of Traffic Incidents. Big Data and Cognitive Computing, 7(1), 22. https://www.mdpi.com/2504-2289/7/1/22

11. Xu, D., et al. (2023). Traffic-ConvLSTM: Urban Traffic Anomaly Detection. ISPRS International Journal of Geo-Information, 13(10), 351. https://www.mdpi.com/2220-9964/13/10/351

12. Saleh, K., et al. (2021). Vision-Based Traffic Anomaly Detection Using Deep Learning and Decision Trees. CVPR Workshops. https://openaccess.thecvf.com/content/CVPR2021W/NTIRE/html/Saleh_Vision-Based_Traffic_Anomaly_Detection_Using_Deep_Learning_and_Decision_Trees_CVPRW_2021_paper.html

13. Gao, H., et al. (2024). Traffic Flow Prediction in Hong Kong Using CNN-LSTM. Machine Learning and Data Mining Journal. https://link.springer.com/article/10.1007/s42421-024-00112-2

14. Chen, Z., et al. (2022). DSGCN for Urban Congestion Forecasting. Information, 14(2), 108. https://www.mdpi.com/2078-2489/14/2/108

15. Quispe, H. G., et al. (2023). Intelligent Traffic Management System in Lima Using ML Algorithms. ResearchGate. https://www.researchgate.net/publication/391712438

16. Rani, D., & Ramasamy, V. (2024). IAOADL-TCC: Intelligent Adaptive Optimization for Congestion Control. Ain Shams Engineering Journal. https://www.sciencedirect.com/science/article/pii/S1110016824006823

17. Ghosh, A., & Basu, S. (2024). Survey on Machine Learning for Traffic Management. Engineering Applications of Artificial Intelligence. https://www.sciencedirect.com/science/article/pii/S0952197624003051

18. Karami, A., et al. (2023). An Autonomous Agent for Flow Anomaly Detection in Urban Traffic. Transportation Research Part C: Emerging Technologies. https://www.sciencedirect.com/science/article/pii/S0968090X23000785

19. Patel, A., et al. (2024). AI-Based Traffic Management System Using YOLOv5 on Edge Devices. ResearchGate Preprint. https://www.researchgate.net/publication/389363577

20. Zhao, H., et al. (2025). Real-Time Traffic Jam Detection Using ResNet on Road Cameras. Scientific Reports, Nature. https://www.nature.com/articles/s41598-025-97942-z

21. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.

22. Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*.

23. Jocher, G., Chaurasia, A., & Qiu, J. (2023). YOLO by Ultralytics. *GitHub repository*. https://github.com/ultralytics/ultralytics

24. Ultralytics. (2024). YOLOv11: Next-Generation Real-Time Object Detection. *Ultralytics Blog*. https://ultralytics.com

25. Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 618–626.

26. Muhammad , S., & Yeasin, M. (2020). Eigen-CAM: Class Activation Map using Principal Components. *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 3504–3513.

27. Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4), 427–437.

28. Everingham, M., et al. (2010). The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88, 303–338.

# 11 Appendix

Appendix A: Ethics Review


## Certificate of Ethics Review

**Project title:** AI-Powered Smart Traffic Management System

| **Name**: | Omar Masoud | **User ID**: | 2270587 | **Application date**: | 19/06/2025 11:10:58 | **ER Number**: | TETHIC-2025-111337 |
|---|---|---|---|---|---|---|---|

You must download your referral certificate, print a copy and keep it as a record of this review.

The FEC representative(s) for the **School of Computing** is/are **Elisavet Andrikopoulou, Kirsten Smith**

It is your responsibility to follow the University Code of Practice on Ethical Standards and any Department/School or professional guidelines in the conduct of your study including relevant guidelines regarding health and safety of researchers including the following:

- University Policy
- Safety on Geological Fieldwork

It is also your responsibility to follow University guidance on Data Protection Policy:

- General guidance for all data protection issues
- University Data Protection Policy

Which school/department do you belong to?: **School of Computing**
What is your primary role at the University?: **Postgraduate Student**
What is the name of the member of staff who is responsible for supervising your project?: **Gelayol Golcarenarenji**
Is the study likely to involve human subjects (observation) or participants?: No
Will financial inducements (other than reasonable expenses and compensation for time) be offered to participants?: No
Are there risks of significant damage to physical and/or ecological environmental features?: No
Are there risks of significant damage to features of historical or cultural heritage (e.g. impacts of study techniques, taking of samples)?: No
Does the project involve animals in any way?: No
Could the research outputs potentially be harmful to third parties?: No
Could your research/artefact be adapted and be misused?: No
Will your project or project deliverables be relevant to defence, the military, police or other security organisations and/or in addition, could it be used by others to threaten UK security?: No
Please read and confirm that you agree with the following statements: I confirm that I have considered the implications for data collection and use, taking into consideration legal requirements (UK GDPR, Data Protection Act 2018 etc.), I confirm that I have considered the impact of this work and and taken any reasonable action to mitigate potential misuse of the project outputs, I confirm that I will act ethically and honestly throughout this project

### Supervisor Review

As supervisor, I will ensure that this work will be conducted in an ethical manner in line with the University Ethics Policy.

Supervisor comments: **No human or animals are involved and an open source dataset will be used.**

Supervisor's Digital Signature: **gelayol.golcarenarenji@port.ac.uk** Date: **19/06/2025**


# School of Computing
# Postgraduate Programme

MSc in Artificial Intelligence and Machine Learning

# Project Specification

**Omar Tarek Elsayed Masoud**

Project Specification

# 1. Basic details

| | |
|---|---|
| Student name: | Omar Tarek Elsayed Masoud |
| Draft project title: | AI-Powered Accident Detection System |
| Course and year: | MSc in Artificial Intelligence & Machine Learning, Year 1 |
| Client organisation: | University of Portsmouth |
| Client contact name: | University of Portsmouth |
| Project supervisor: | Dr. Gelayol Golcarenarenji |

# 2. Outline of the project environment

The client is the University of Portsmouth. The aim of this project is to address the growing demand for real-time accident detection in urban environments using CCTV footage. Current monitoring systems rely heavily on human operators, which makes them slow, inconsistent, and prone to error when identifying critical road accidents. This leads to delayed emergency response, increased congestion, and higher risks to public safety.

The proposed system focuses on building an AI-powered accident detection pipeline capable of distinguishing accident and non-accident scenes, localising accident regions, and providing interpretable outputs. This will support traffic management stakeholders and emergency responders by automating monitoring tasks and improving efficiency, safety, and decision-making.

# 3. The problem to be solved

Manual monitoring of CCTV for accident detection is inefficient and error-prone. There is no scalable solution to automatically identify accident events in diverse road and weather conditions.

This project aims to develop an AI-powered accident detection system that uses computer vision and deep learning to:

- Detect and classify accident vs. non-accident scenes.
- Localise accident areas with bounding boxes.
- Ensure results are explainable to human operators.
- Operate in near real-time to support emergency response.

**Objectives:**

- Process CCTV footage to classify accident vs. non-accident scenes.
- Localise accident regions using bounding box annotations.
- Implement a CNN pre-filter to reduce computational load.
- Train and compare YOLOv8 and YOLOv11 for accident detection.
- Provide explainability through Grad-CAM/EigenCAM visualisations.
- Evaluate models using metrics such as accuracy, precision, recall, F1-score, and mAP.

**Drawbacks:**

- Large variation in CCTV data (lighting, weather, occlusion) may impact detection accuracy.
- Risk of false positives and false negatives in critical accident cases.
- Real-time inference requires efficient optimisation.

# 4. Breakdown of tasks

- **Approach:** Iterative development using CRISP-DM, covering preprocessing, modelling, evaluation, and deployment.
- **Research:** Review accident detection studies using CNNs, YOLO variants, and explainability methods.
- **Potential Tools:** Google Colab, TensorFlow/Keras, PyTorch, Ultralytics YOLO, OpenCV.
- **Skills needed:** Knowledge of deep learning, computer vision, object detection, model optimisation, and explainability techniques.
- **Build:** Accident vs. non-accident CNN classifier, YOLOv8 and YOLOv11 accident detectors, bounding box overlays, Grad-CAM explainability, evaluation pipeline.

# 5. Project deliverables

- AI-based accident detection system (trained CNN + YOLO models).
- Visual outputs with bounding boxes highlighting accident areas.
- Explainability results (EigenCAM heatmaps).
- Evaluation metrics and comparison tables (precision, recall, F1, mAP).
- Final written report and project documentation.

# 6. Requirements

The system must:

- Accurately detect and classify accident vs. non-accident frames.
- Localise accident regions using bounding boxes.
- Provide interpretable outputs to support operator trust.

- Operate in near real-time under varied conditions.

Requirements will be validated iteratively with supervisor input.

# 7. Legal, ethical, professional, social issues

- **Legal:** Must comply with GDPR and data protection laws. Only anonymised or public datasets will be used.
- **Ethical:** Avoid false alarms that could waste emergency resources. Provide explainability for transparency.
- **Professional:** Ensure results are reproducible, fair, and aligned with AI best practices.
- **Social:** Enhance road safety, reduce accident response times, and minimise human monitoring burden.

**Ethics Approval:** A publicly available Kaggle dataset was used to avoid handling personal data.

# 8. Facilities and resources

- **Computing:** Google Colab Pro with A100 GPU; personal laptop (16GB RAM).
- **Resources:** Kaggle Accident Dataset, publicly available annotations.
- **Constraints:** Limited Colab GPU time may restrict batch size and training epochs; optimisations required.

# 9. Project plan

**Phase Timeline:**

- Weeks 1–2: Literature review, requirements definition.
- Weeks 3–5: Dataset acquisition, cleaning, and preprocessing.
- Weeks 6–9: CNN accident classifier development.
- Weeks 10–13: YOLOv8 training and evaluation.
- Weeks 14–16: YOLOv11 training and evaluation.
- Weeks 17–18: Integration of CNN + YOLO pipeline.
- Weeks 19–20: Evaluation, error analysis, documentation, submission.

**Risks and Mitigation:**

- **Data imbalance:** Use augmentation and resampling strategies.
- **Model overfitting:** Apply dropout, early stopping, and validation monitoring.
- **Time constraints:** Maintain backups and regular reviews with supervisor.

# 10. Supervision meetings

Meetings will be held weekly or biweekly on Zoom. Communication through email will provide regular progress updates. Supervisor input will be used to refine requirements and validate results.

# 11. Project mode

| | | |
|---|---|---|
| Registration mode | Full Time | |
| Project mode | Full Time | |
| Planned submission deadline | 11/9/2025 | |

# 12. Signatures

Signature: Date:

| | | |
|---|---|---|
| Student | Omar Masoud | 22/05/2025 |
| Client | | |
| Project supervisor | | |