Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Deep Learning–Driven Traffic Detection and Flow Optimization using Simulation-Based Analysis in Spatial Domain

MANISHA AERI
Graphic Era University
SIMAR SINGH RAYAT
rayatsimar@gmail.com

Graphic Era Hill University
SUJAL THAPA
Graphic Era Hill University

Research Article

Keywords: Traffic detection, synthetic simulation, YOLOv5-S, spatial features, adaptive signal control, queue modeling, feature extraction, pre-processing, Poisson traffic model, traffic optimization, vehicle classification

Posted Date: February 25th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-8912354/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Abstract

The paper examined an entire system for recognizing traffic and optimizing green signal timing with artificial video data as actual videos were difficult to utilize. This research employed pre-processing, feature extraction, YOLO detection, and then simulated signal controls to determine waiting times. Results demonstrated improvements in detection accuracy and queue reductions during heavy traffic periods. The system may not be optimal; however, it ran consistently well throughout the majority of the test scenarios. In addition to demonstrating the feasibility of this approach (combining machine learning, deep learning, and simulation) to develop a traffic management concept which performs better than traditional fixed-time traffic signal systems, this paper also aimed at creating a relatively simple methodology so that it could run on low-end hardware configurations.

1 Introduction

As traffic management is currently affected by traffic congestion, unpredictable behaviors of vehicles, and poor signal synchronization, traffic automation is recognized as an important research trend. Conventional traffic management was usually based on fixed time based signaling or manual operation or a rudimentary type of sensor feeding; hence these systems have a constraint in responding to sudden spikes in traffic volumes. As a result of urbanization and growing private automobile ownership, modern traffic management infrastructures lack the capacity to respond fast enough to change in traffic conditions. This deficiency often leads to stretching of queue and extended periods of driver's stay in idle states that contribute to fuel wastage and increased air pollutant emissions [1]. Recent investigations have therefore established their direction to dynamic, scalable solutions which are based on data-driven methodologies, rendering themselves able to automate the traffic management. Proper implementation of machine learning (ML) and deep learning (DL) methods makes it possible to convert the raw traffic video streams into meaningful information to be used in the real-time control of traffic signals. The ability to obtain real-time operable information from video feed has given rise to significant interest in the research for developing automated traffic management pipeline including vision-based detection, the analysis of the spatial domain and simulation-driven optimization [2].

The Venn diagram describes the conceptual chain of development of traffic perception systems from traditional, feature-engineered vision methods to an infinite variety of advanced intelligent transportation analytics. Early methods of machine learning were based on handcrafted descriptors and statistical tracking, where the objects were characterized by measurable visual features instead of representations learnt from them. In comparison, the deep learning research is a good example of the emerging data-driven perception using convolutional and transformer-based detectors for direct assimilation of hierarchical features from raw images. The transportation engineering field is represented by models that analyses the flow dynamics without the visual recognition. The intersectional pairs represent transitional periods in research. The combination of machine learning and deep learning produces the hybrid systems that combine engineered and learned features. The intersection between machine learning and traffic modelling represents first intelligent systems that are used in traffic estimation,

based on the motion analysis and the use of rules obtained by one camera. The intersection of deep learning and the traffic plays corresponds with the modern perception-based systems, like the YOLO-based vehicle detection, lane detection, violation detection, aerial observation of traffic. The central intersection represents the current state-of-the-art in which the outputs of the detections are not final goals, but inputs into a higher-level reasoning. In this filed, spatiotemporal modelling, graph-based road network analysis, multisensor fusion, predictive risk analyses are involved in transmutation of visual detection into informative action to traffic scene understanding. Consequently, installed at the figure's foreground is the shift from primitive vehicle recognition to the complete interpretation of transportation dynamics, as shown in Fig. 1.

The recent advances in both ML and DL have enabled the performance of object detection and classification performance for traffic related applications to improve significantly. Early methods in ML used hand-craft feature extraction techniques like Scale-Invariant Feature Transform (SIFT), Histogram of Oriented Gradients (HOG), Local Binary Patterns (LBP), Gabor filter and edge descriptors for the purpose of object identification in the images. While these techniques had relative success, they faced limitations in the presence of occluded objects, variable illumination and dense traffic cases. The advent of DL made it possible to deploy architectures, such as YOLO, Fast R-CNN, SSD, VGG, ResNet, and EfficientNet as robust and high throughput architectures for detecting vehicles and pedestrians. These models are capable of analyzing the frames of a video in a very fast manner and producing precise bounding box predictions. As opposed to earlier ML paradigms, the deep neural networks are able to extract features that are useful for discriminations by themselves without the requirement for custom, hand-crafted descriptors.

Nevertheless, the creation of such networks requires large datasets and high-quality annotations as well as constant preprocessing before the real-time stream of traffic can be analyzed. Furthermore, while DL has produced remarkable advances in visual object detection, conventional ML techniques still have their place in scenarios were relying on a low compute simulation [3], a limited number of resources is available, or where the analysis of traffic over previously extracted features in the spatial domain is of interest. Feature extraction in the spatial domain relies on intimate knowledge of the vehicular flow physical characteristics while driving in dynamic scenes [4]. The most majority of scholarship is on the frequency domain feature extraction or only the deep feature extraction; however, the spatial domain techniques provide some other advantages in terms of control granularity and interpretability. Spatial domain attributes include texture variation, intensity gradients [5], contour structures, lane density, and per-pixel segmentation map features, all of which can be exploited to determine the nature of the traffic movement in the particular intersections or parts of the roadways. Such features have been found to be especially useful in scenarios where low light or blurred imaging conditions are common or if there are scenarios where preliminary processing such as noise reduction, sharpening of images and extraction of the region of interest (ROI) has to be performed before the data is placed into an ML or DL model.

By leveraging the spatial information, hybrid detection systems can be designed, in which the traditional ML approaches are combined with DL models [6] or vice-versa, thus limiting the computational overhead

and increasing the reliability while being simulating friendly, with the pipelines being implemented in either the MATLAB, python, or NS-3 software [7]. Despite great progress in working on the detection and management of traffic, there are prominent gaps in the existing literature. Many researchers focus on the study of vehicle detection accuracy and do not consider the problem of real-time adaptability for traffic signal control [8]. Other investigations develop the methods by using large-scale real-world data sets but do not devise means to generate the real-world data sets in case of unavailability of large real-world datasets.

The work process in the work flow of the YOLO-based traffic monitoring system is shown in the figure below, which starts from the raw video input and ends with definitive results of the semantic detection output. A video frame of traffic is resized and normalized to obtain a standardized illumination and spatial scale, and then the frame is divided into a spatial grid of S*S size. The processed frame is fed to a deep convolutional backbone, to make the hierarchical feature extraction, which is followed by a feature pyramid neck to aggregate multiscale contextual information indispensable to detect the objects under variable scales. The detection head returns bounding box, confidence and class probability for every grid cell which is the raw prediction tensor. Given the likelihood of multiple detections of the same object, a post-processing stage is used in which confidence thresholding and non-non maximum suppression are used to eliminate redundant bounding boxes. The final output is labelled entity objects of interest in traffic - cars, buses, pedestrians and traffic signals-hence clearing the way for traffic related road scenes to be interpreted in real-time. The pipeline is a typical example of the methodology of modern object detection systems, where pixel-level information is transformed into structured knowledge about the transportation by mechanisms of spatial division, deep feature learning, and a probabilistic filter in the complex architecture, as shown in Fig. 2.

Detecting and monitoring traffic using isolated deep learning models has been applied by several scholars; however, very few are going further, to optimize traffic flow by implementing. Moreover, there are very few of them that offer a holistic pipeline leading from raw traffic video acquisition, through preprocessing, feature extraction, ML and DL detection [9], to the final optimization using simulation tools. No comparative study is done, to date, to assess the performances of hand-crafted spatial-domain features versus deep learning features in simulated environments. Finally, no study provides a realistic roadmap for the implementation of the proposed approach including the preparation of datasets, training and validation of their models and their integration with adaptive signal control algorithms tested in a controlled simulation environment [10]. This article overcomes the above-mentioned shortcomings by proposing an end-to-end, simulation-based methodology for automated traffic detection and optimized traffic flow control. The contribution starts with the creation of an artificial or real dataset in which vehicular behaviors emulation is conducted in a range of density levels, lighting conditions and intersection types.

In case perhaps real data sets are lacking, a synthetic data set sourced from traffic simulators or controlled video feed brings the needed data variability for the pipeline. The mid-level data generation process includes extracting frames of traffic videos, annotation of the vehicles and pedestrians, image

size reduction to standard size and saving metadata of lane occupancy and vehicle typology. The dataset informs the later research phases that ensure that developed pipeline is independent of external datasets and will be able to be dropped into place for optimizing traffic flow for different roadway configurations [11]. The paper presents a novel approach to preprocessing in which a pipeline in order to improve pipeline robustness before features extraction. Noise-Reduction philters help you to get rid of extraneous artefacts, while ROI extraction helps to separate lane level information from extraneous data. Techniques for frame enhancement in low light condition [12] along with morphological operations for enhancing edges of the vehicles are also incorporated. These preprocessing steps ensure that there are no inconsistencies in the output despite weather conditions, shadows, or night conditions, or the resolution of the source video, and thus ensure the consistency of the further processing stages in ML and DL classify the images [13].

A further main contribution is in the field of the feature extraction and selection. Rather than pure deep learning feature extraction procedures, the research combines space domain extraction methods (LBP, HOG, gradient maps, edge detection) to provide interpretable structural data [14]. Experimental evaluation of the discriminative ability and the computation expenses lead to the design of a hybrid object recognition framework which achieves a balance between accuracy and speed. Training both ML methods and DL architectures on the selected feature set allows the cross comparison between methods through a similar simulation pipeline to evaluate the impact of the features included on the detection accuracy and the processing time to enable real-time traffic analysis [15]. Depending upon the complexity of the given simulation the detection module selects the ML or DL approaches. High precision multi object detection uses DL models (YOLO, Faster R-CNN) while classification problems use the lightweight ML algorithms (SVM, Random Forest) based on spatial features [16].

The complexity scale in the framework of detection; Modular detection allows upscaling of computation with the used simulation platform. Python is used for DL experiments, and MATLAB for feature extraction and image preprocessing while NS-3 is used for traffic flow modelling making the system versatile on different platforms [17]. The most important contribution is related to traffic flow optimization after they are detected. Detection result-related vehicle counts, density measures, and lane occupancy information generated by detection as input for an adaptive signal control model. Unlike fixed-time, the adaptive model is a dynamic model which adjusts the duration of green and red phases based on the real-time traffic conditions, which will reduce the total waiting time at the intersection and make the traffic flow smoother running. Simulation based validation is used to validate the adaptive model under various traffic conditions and load intensities [17]. Such simulations allow researchers to measure the performance of a system under a controlled environment, eliminating the need for deployment in a field setting and allowing them to study the limits of a system.

Consequently, with subsequent improvements in detecting accuracy, performing traffic system optimization practically and directly applicable towards smart cities as the augmentation of its data set preparation and advanced pre-processing, spatial domain feature extraction, hybrid ML/ DL detecting,

adaptively traffic signal simulating, the paper provides a complete and reproducible structure for future scholarly inquiry.

2 Literature Survey

Automated traffic management technology has been developed considerably over the past ten years due to increasing demands for improved urban mobility, reduced traffic congestion and minimized traffic impacts to the environment. Traditionally traffic management systems have utilized induction loop detectors, infrared detectors and manually observing traffic flow [18]. These types of traffic management systems were relatively inflexible and provided very little contextual data. In early efforts using computer vision to analyze video traffic data [19], researchers used background subtraction and frame differencing to identify moving vehicles. Unfortunately, these initial attempts at using video analysis were also heavily dependent upon consistent lighting and minimal camera vibration; therefore, did not provide reliable results across all conditions [20]. As computer vision continued to develop, researchers began investigating more robust feature-based methodologies that ultimately led to the development of machine learning and deep learning methodologies currently being used for traffic analysis today [21].

The first applications of machine learning to traffic analysis typically employed hand-designed features from the spatial domain. The most popular techniques used to extract features from traffic images include Histogram of Oriented Gradients (HOG), Local Binary Patterns (LBP), Scale-Invariant Feature Transform (SIFT), and Gabor filters [22]. HOG was originally developed by Dalal and Triggs to detect pedestrians in images, and their methodology was then applied to a variety of vehicle detection research studies to analyze structural features in traffic images (edge orientation and gradient intensity) [23]. While many researchers successfully combined the features mentioned above with classification algorithms (e.g., SVM and Random Forests) to achieve reliable vehicle detections, these algorithms failed to perform well when detecting vehicles within cluttered environments, in cases of occlusions, in cases of varying vehicle sizes, and in cases where there are inconsistencies in lighting (especially nighttime traffic) [24].

To resolve some of the above-mentioned constraints, deep learning has received considerable interest in the area of traffic analysis with the help of Convolutional Neural Networks (CNNs). By using CNN's, researchers were able to use an automated process to learn hierarchical features from their data; therefore, eliminating the requirement for researcher-created feature descriptors. Region Proposal Network (RPN) based Faster R-CNN has been successful in detecting vehicles at a high level of accuracy [25]. Single Shot Detectors (SSD), such as YOLO and SSD, are faster than traditional methods of detection by turning detection into a regression problem. YOLOv3, YOLOv5, and other versions of YOLO [26] have become one of the most popular forms of research in traffic detection due to their ability to achieve high speed detection along with high-quality detection precision. Many studies have shown that these models are able to detect many different vehicle types including cars, trucks, buses, and two-wheeled vehicles under various adverse environmental conditions [27]. However, it is critical to note that

deep learning models typically require large amounts of training data and the quality of the models will degrade when they receive noisy frames, shadows, fog, or low-resolution traffic video [28].

Researchers have also studied hybrid approaches that integrate spatial domain features with features that are learned using deep learning models. Examples include integrating LBP (Local Binary Patterns) texture-based features within CNN [9] pipelines to increase robustness to low-contrast images [29]. Researchers have also utilized pre-processing techniques, such as histogram equalization, Gaussian filtering, and morphological operations to improve the quality of input frames before passing them to deep learning models [30]. These hybrid approaches reduced the computational costs associated with processing images using deep learning models, improving generalization and enabling researchers to develop resource aware traffic analysis systems that may be run on edge devices, which may be located near intersections [31].

Traffic flow estimation, and smart signal optimization are also common topics of study. The classical traffic flow theories, such as the Webster Formula and the Highway Capacity Manual, were static formulas for signal timing, which did not account for real time variations. Intelligent signal control methodologies began to be developed that utilize the outputs of vision-based systems [32]. There has been considerable interest in utilizing reinforcement learning methods, specifically Q-Learning and Deep Q-Networks (DQN) [33], to optimize the duration of red and green cycles based upon the length of queues of vehicles. While these methodologies have shown promise in simulation, most of them require substantial state space exploration, and performance is highly dependent upon accurate real-time detection of vehicles [34].

Simulated environments were used to validate much of the traffic control research. Simulators such as SUMO, NS-3 and MATLAB Sim Events allowed researchers to develop scalable models of traffic intersections, vary the rate at which vehicles arrive at the intersection, and evaluate signal optimization methodologies without requiring actual deployment. Many studies have combined simulated traffic detection modules with SUMO or NS-3 to investigate the effect of detection accuracy on queue lengths, travel times and through put of the intersection. A significant shortcoming of the existing body of work was that few studies investigated the workflow of detecting vehicles and optimizing signals in a single continuous process, from creating the dataset, to preparing it for processing, extracting features, detecting vehicles, and finally, adjusting the signals [35].

The existing body of literature also demonstrates a large deficiency in terms of available datasets of traffic under a variety of real-world conditions. While there are datasets available, such as UA-DETRAC, City Flow, and KITTI, they do not always represent the specific traffic characteristics of different regions, nor do they always contain high quality video feed, nor do they always include a variety of environmental conditions [36]. As a result, several authors proposed the use of synthetic traffic data created by simulating traffic using simulators or three-dimensional rendering engines to complement real datasets. The synthetic data provides a method to avoid the expense of manually annotating the data [37], and can provide controlled variations in density, weather, and angle of the cameras.

Overall, the body of literature indicates rapid growth in the development of new algorithms for detecting vehicles, new pre-processing methods for preparing the images for analysis, and new methods for controlling traffic signals. However, relatively few studies have demonstrated a complete pipeline for managing traffic from preparing the dataset to preparing the images for analysis to detecting the vehicles to adjusting the signals. Most studies focused solely on increasing the accuracy of vehicle detection and have paid little attention to how those detected vehicles affect the subsequent traffic optimization [38]. This shortcoming creates a compelling argument for developing a comprehensive solution that includes preparing the dataset, preprocessing the images, integrating spatial information into the features extracted from the images, and using a combination of machine learning and deep learning methods to detect vehicles, and then simulating traffic to continuously adapt the signal timings the type of solution being addressed by this study.

3 Problem Statement and Research Objective

3.1 Problem Statement

Urban intersection traffic congestion is continuously increasing with the rapid increase in the number of vehicles on the road; and the inconsistency in how drivers operate their vehicles, along with inefficiency of current fixed timing traffic signals. The majority of current traffic management systems currently lack the needed intelligence to adjust their timing according to changing conditions, in real time. While there has been an increase in the use of camera-based monitoring technology, the vast amounts of data collected via this means have gone unused since most conventional systems cannot convert the unprocessed video footage into intelligent decision-making inputs. As far as prior research has come in terms of improving the accuracy of detection capabilities, many of those studies have not provided a unified process or pipeline that includes dataset creation, robust preprocessing, spatial domain feature extraction, machine learning/digital learning (ML/DL) based detection, and final signal optimization. Additionally, obtaining consistent real world traffic videos can be challenging, which is why simulated datasets are crucial for generating synthetic datasets [39]. Therefore, this study will address these deficiencies by developing a comprehensive, simulation-generated traffic detection/flow optimization system utilizing simulation-based tools such as MATLAB, Python, NS-3, and controlled simulation environments.

3.2 Dataset Preparation using Synthetic Tools

This study's primary objective is to create a controllable and malleable dataset using synthesized tools (as opposed to using only real-world traffic videos). The majority of current simulation tools and video generators are built in either MATLAB or Python based engines, or 3D traffic simulators that can generate video of realistic traffic scenarios based on the parameterization of factors such as traffic density, weather, time of day, and types of vehicles present. After generating these synthetic videos, they are broken down into frames and each frame is labeled to identify vehicles such as cars, buses, trucks, and motorcycles. The use of synthetic data has many advantages including consistency; eliminating the

high cost associated with collecting real world video; allowing multiple runs of an experiment to occur under the same conditions for accurate evaluation of models.

3.3 Pre-processing of Simulated Data

A second objective aims at improving the quality of synthetic images by applying a series of systematic preprocessing steps prior to the generation of synthetic image frames. These steps include removing noise in generated images with methods such as gaussian filters, median filters, or optimizing contrast to improve visual quality. In addition, processing regions of interest (ROI) to focus only on specific sections of the roadway (i.e., specific lane lines or road segments) will reduce computation time and memory needed. Uniform frame size for each model that utilize machine learning (ML)/deep learning (DL) will be achieved by resizing the frames. Synthetic data generally contains less noise than real world data; however, additional preprocessing is typically necessary to introduce variability in simulated conditions (e.g., reduced illumination and/or shadow interference), resulting in a dataset that can effectively train detection models that are robust in their ability to detect objects under various conditions.

3.4 Feature Extraction and Spatial Domain Selection

An important goal of the research presented here will be to take useful feature characteristics out of a synthetic frame (for example LBP, HOG, edge maps, etc.). The spatial domain features that we have selected are all related to different types of visual cues, including texture, contours and motion; these are all very important for recognizing vehicles. While deep neural networks are able to learn features without prior knowledge of what those features are, when they are combined with a set of spatial domain features, they improve the interpretability of the model and reduce computational cost. A second benefit of feature selection is that it allows the identification of the best descriptor for each task or problem so that the system can run efficiently given the limitations of the simulator.

3.5 Real-Time Detection and Classification Model

The next goal will be to create a real time detection and identification of vehicles (counts) from the synthetic video. The above-mentioned deep learning algorithms (YOLO or Faster R-CNN), have shown good performance with high accuracy in detecting objects, but they require significant processing resources. The lightweight machine learning algorithms are much less resource intensive and can run significantly faster than the deep learning architecture. This approach also enables you to train your model using synthetic data and apply it to dynamic simulations performed within MATLAB, Python, or NS-3. In this way, you avoid the need to collect large and expensive real world data sets.

3.6 Traffic Flow Optimization through Simulation

Ultimately, this project will develop an intelligent traffic flow optimization model capable of utilizing detection information for the dynamic adjustment of signal timing. Simulations are generated using simulated intersections developed with MATLAB's SimEvents or NS-3 mobility models or with Python-based traffic generation tools; these simulations permit the collection of data regarding the length of

queues, the average amount of time vehicles spend stopped (delay), and the total amount of time vehicles spend stopped (halt time). An optimization algorithm dynamically adjusts the duration of green and red signals as a function of vehicle density per lane. The use of simulation-based adaptive control demonstrates how the movement of vehicles through an intersection can be made safer, faster, and more efficiently than by comparison to traditional fixed-timing signal systems.

4 Methods

The Fig. 3 shows the envisaged end-to-end traffic intelligence architecture where the perception, analysis and control are synthesized in a unified framework via a feedback loop. Synthetic and real traffic video data are extracted and pre-processed at first to reach the resolution standardization and noise attenuation. For feature extraction spatial features are collected by convolutional neural networks and edge-based methods while feature selection and fusion provides to combine complementary information. The function of the detection module is to perform object recognition and object tracking to produce structured results describing the existence and movement of vehicles. Subsequent post-fire processing is carried out to remove redundant predictions, and the system passes the detected information-on in the form of traffic flow parameters (vehicles counts and speed estimation)-in an adaptive traffic signal optimization module. Within this component, policies for the signal timing are modulated by reinforcement learning agents which take past traffic conditions into consideration [40]. A queue-simulation module evaluates delay and congestion behavior and performance metrics from the queue are passed back to the control loop. Evaluation and ablation analyses are used to link the performance of the system, and the visualization components are used to generate interpretable traffic heatmaps and analytical plots. Accordingly, the framework goes beyond the perception by making it part of a decision-making cycle, thus allowing not only observation in real time but also monitoring and dynamic traffic control.

The Fig. 4 presents a stratified taxonomy that shows the process of transforming heterogeneous data of a traffic and gradually turning them into structured detection targets for running a YOLO system. The pyramid structure gives a focus on abstraction: raw observational diversity occurs in lower tiers, while mathematical representations occur in compressed upper tiers that are used in the process of learning models. The bottom level lists sources of data acquisition, including both real and synthetic collections (e.g. OpenCV, KITTI, COCO, BDD100K, Cityscapes, dashcam videos, uav aerial overflights, road sidedes CTVs, and driving simulator environments). These sources are sources of variability in viewpoint, resolution, illumination and traffic behavior, thus enabling robust model generalization. The following tier is used to define scene domains, where objects exist in the context of an environment: streets of cities, highways, intersections, parking areas, nights, unfavorable weather conditions, aerial views, and lateral-acquired roadside cameras [41]. This level differentiates the characteristics of the physical environment from the source of raw data. Above this, the object classes of the semiotics represent the semantic entities of interest - cars, buses, trucks, motorcycles, bicycles, pedestrians, traffic lights, traffic signs, and lane markings.

This layer adds semanticity by transmuting visual patterns into elements of the transport relevant for transportation. The next level defines that annotation schema, coding for bounding boxes, categorical coding, tracker code, speed estimation coding, and coding of occlusion. Visual semantics therefore become machine readable supervision signals that are used during the training. At the top of the pyramid is the detection-output representation, which describes the internalization of the knowledge annotated, through the learning algorithm. Each object is represented using normalized spatial coordinates, confidence estimates, and class probabilities vectors. So doing, this abstraction reduces complex real world traffic environments into numerical summaries that can be optimized for the applications of real time inference [42]. Overall, the pyramid shows the path from heterogeneous sensory data to minimal mathematical representations to transfer the path from the observation of the environment to the probabilistic detection under YOLO learning algorithm.

4.1 Dataset Creation in Synthetic Environment

The study is based upon a completely simulated data generation because it is difficult to obtain real-time video for each traffic situation, as well as there are many legal restrictions and complications that occur when trying to do this. As a result, the dataset has been generated utilizing synthetic tools such as a MATLAB Traffic Generator, Python simulation script, and CARLA Simulator's simple environmental scene. The synthetic traffic videos were generated with multiple features (density, weather, random angles, etc.) as well as the ability for lane changing behavior to keep the model from being over-trained and only able to identify one specific sample type.

Each video had been decomposed into frames by using Python OpenCV, where the frame extraction rule was $f(t) = V(t \cdot \Delta)$with $\Delta = 1/10$seconds so that there were 10 fps data. Following frame extraction, we manually annotated the object classes (car, truck, bus, bike and pedestrian) by hand utilizing makesense.ai, and saved the annotations to our database as JSON files; every frame was also tied to some metadata (such as time stamp, which lane it was on, total vehicles on that lane, and direction of movement) that was determined from a simple optical flow displacement equation:

$$d = \sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2}$$

1

so that direction classification becomes steadier during synthetic testing.

4.2 Pre-Processing of Synthetic Frames

The synthetic frames had been passed through pre-processing steps so that the training don't break due to noise or blur. Though synthetic data is cleaner, still the paper had introduced artificial noise to make model more robust. Gaussian noise had been added using:

$$I_{noisy}(x, y) = I(x, y) + \mathcal{N}(\mu, \sigma^2)$$

2

Then again it was filtered with a Gaussian filter to simulate conditions found in the real world. The Region of Interest (ROI), i.e., road area was masked by using color thresholding in HSV space, thus eliminating unnecessary backgrounds. In addition, frame size reduction to 640\times640 was used as the chosen algorithm family is most consistent when working with squared shapes. Therefore, all the steps above have been used to improve the consistency among the simulated clips.

4.3 Feature Extraction and Spatial Domain Selection

In this part, spatial domain features had been extracted because this study wanted little bit control on model rather than only deep learning. So, features like Histogram of Oriented Gradients (HOG) had been computed by calculating gradient magnitude:

$$G = \sqrt{G_x^2 + G_y^2}$$

3

and gradient orientation:

$$\theta \ = \tan^{-1}(G_y/G_x)$$

4

Features for both the pixel blocks of images and LBP features were created to better work well in synthetic low texture surfaces. Spatial features were then combined with deep features to provide a hybrid feature set that was used by the model. The Variance Threshold approach was also utilized for selecting which features to use; the model was allowed to remove any feature with a variance of less than 0.001. This process provided a method for reducing the number of dimensions as well as providing a way to speed up the time it takes to train the model.

4.4 Detection Algorithm Choice and Why It Had Been Used

The research had identified YOLOv5 to be the main detection algorithm to be implemented within the paper. This had primarily been due to the fact that the most important factor when working in a simulation environment is real-time processing speed rather than high levels of precision. YOLO was able to process full frame images in a single pass (i.e., single shot), which provided an advantage in simulated traffic environments where shapes and sizes remained unpredictable. The other models examined (e.g., Faster R-CNN) were found to be slower; the multi-stage pipeline also caused long simulation cycles. YOLOv5 utilized an anchor-based bounding box regression using the following equation:

$$L = L_{cls} + L_{obj} + L_{bbox}$$

5

where L_{bbox}had been IoU loss that stabilised box prediction.

4.5 Model Design, Training and Optimization Strategy

The YOLOv5-S model architecture had been created based on its small simulation environment and the backbone CSPDarknet had been trained using augmented synthetic images in the form of flips, hue jitter, and scaling. It had been trained for 120 epochs at a batch size of 16 and used SGD as the optimizer with an initial learning rate of 0.01 and momentum of 0.937. In addition, it had been monitored to see if there was any overfitting of the data during training. The problem had been defined as a multi-class detection problem that is supervised; therefore, each video frame had labeled pairs (x_i, y_i), which represented object class and bounding box. The goal of the training process was to minimize the total amount of loss of the detection system. Additionally, the authors did a very limited ablation study to demonstrate the impact of removing the spatial features to determine their contribution to the overall accuracy of the system [43]. As expected, when the authors removed the spatial features, they saw a drop in accuracy which demonstrated that the spatial features were contributing to the stability of the system.

4.6 Simulation-Based Deployment and Testing

After the model had been trained, it had been deployed inside a MATLAB-based synthetic intersection where vehicles had been generated using Poisson arrival model:

$$P(N = k) = \frac{\lambda^{k} e^{-\lambda}}{k!}$$

6

so, traffic be little random like real world. YOLO model had processed each frame and vehicle counts had been fed into a traffic light optimization module. The signal timing rule had been defined as:

$$T_{green} = T_{base} + \alpha \cdot D$$

7

where $D$was detected density. This had minimized halt time.

4.7 Evaluation Metrics and Baseline Comparison

The model had been evaluated using mAP@0.5, precision, recall, and F1-score. For traffic flow, average halt time and queue length had been measured in simulation. Baseline comparison had been done against a fixed-time signal model and also YOLO without spatial features. Ablation results show hybrid features had improved stability especially in heavy density scenes.

5 Simulation Setup

All experimental frameworks were conducted within a Hybrid Simulation Environment created using MATLAB SimEvents, Python-based Synthetic Video Generator, and a regulated CARLA simulation loop. The primary traffic scenario was performed in CARLA due to its ability to provide parametric controls for road topology, lane geometry, vehicle spawn rates, weather conditions, sun heights, and sensor placements. The camera sensor was placed at a height of 7.5 meters with a pitch of -17° to simulate a typical CCTV angle and the render resolution was set to 1280 × 720 pixels. A Poisson distribution λ = 18 vehicles / min had been applied to vehicle arrivals in CARLA to ensure an uneven distribution that would be appropriate for assessing adaptive signals.

The Fig. 5 outlines the experimental framework used to train and test the model of traffic perception in a simulation-oriented framework. A discrete-event simulator, implemented in the computer software's programming language, could the method, by setting pace traffic situations and management settings. These triggers create two streams of complementary data, namely synthetically generated video sequences created by a Python-based rendering module and photorealistic simulation (RGB, LiDAR, radar) sensor observations (and corresponding annotations) captured in the CARLA simulation environment. The two streams are aggregated and standardized inside a preprocessing buffer; they are made into frames with one common format and ready to be fed to a model for training and inference [44]. The processed data go in a YOLO object detection network written in a deep learning framework that produces the detection output, loss metrics, mean average precision and other performance metrics. These results are then visualized through an evaluation dashboard, and at the same time routed back in to the simulation environment. The feedback loop allows iterative changes in scene conditions and control signals to be made in the simulator for a variety of traffic and weather conditions. By connecting perception and the controllable synthetic surroundings, the framework has the ability to validate the models used for traffic detection under reproducible conditions, thus overcoming the disadvantages of static datasets [45].

The synthetic video feed was then processed through a Python pipeline using OpenCV to capture frames at a 10 Frame Per Second (FPS) and pass each frame through the YOLOv5-S inference module. The YOLOv5-S module utilized CUDA-based GPU Kernels to perform real-time processing on each frame. After vehicle counts, lane wise occupancies and direction vectors computed from optical flow had been exported to MATLAB in JSON packets every 100 ms.

MATLAB SimEvents was used to model intersection level behavior. Each lane was modeled as a discrete event queue with service times being equivalent to green light durations. Traffic behavior generated within SimEvents was synchronized with CARLA-generated densities to match visual representations of the traffic. The adaptive signal timing module used the relationship:

$$T_{green}\left(t\right) = T_{base} + \alpha \cdot \rho\ \left(t\right)$$

8

where $\rho\ (t)$represented instantaneous density from YOLO detection. The execution logic had been updated every simulation cycle using event-driven triggers.

Validation of the simulated loop with both systems was done in terms of stability using a synthetic session that ran for 30 minutes at low, medium, and high congestion levels to validate performance metrics (mean halt time, throughput, queue growth) via log extraction from SimEvents and metrics (map, precision, inference latency) from the Python detection engine, all the associated simulation setup step observed are discussed in Table 1.

Table 1
Proposed Hybrid Detection Model Architecture.

Module / Layer	Configuration/ Parameter	Output Dimension	Purpose
Input Frame	640×640 RGB (resized synthetic frame)	640×640×3	Standardize frame resolution for YOLO backbone
Pre-processing Block	Gaussian filtering (σ = 1.2), ROI masking, HSV thresholding	640×640×3	Noise removal, isolate roadway region
Spatial Feature Extraction	HOG (cell: 8×8, block: 2×2, bins: 9), LBP (radius = 1, P = 8)	1,024-d vector	Extract structured texture + gradient patterns
Feature Fusion Layer	Concatenation of spatial features + deep backbone features	1,024 + deep features	Multi-representation input for detector
YOLOv5-S Backbone (CSPDarknet)	Conv → C3 block → Conv → C3 → SPPF	80×80, 40×40, 20×20 feature maps	Hierarchical feature extraction
Neck (FPN + PAN)	Upsample, C3, lateral connections	80×80, 40×40, 20×20 fused maps	Multi-scale feature fusion
Detection Head	Anchor-based bounding box regression, objectness score, class prediction	Vector (bbox + obj + class)	Predict vehicle locations + classes
Loss Function	Total Loss = Lcls + Lobj + Lbbox; IoU-based bbox loss	Scalar	Optimize detection accuracy
Post-processing	Non-Max Suppression (NMS) IoU = 0.5	Bounding box set	Remove duplicate predictions
Output Layer	Class name + confidence + box coordinates	JSON packet	Sent to MATLAB/NS-3 for traffic optimization

6 Results

The whole simulation pipeline functioned as well or even better than anticipated. As expected, the pre-processor greatly enhanced image clarity, as some frames were clearer after removing noise from the original frames. Feature extraction showed that HOG and Sobel performed better in synthetic traffic environments than LBP, although LBP provided additional support and did not fail. The loss functions for all three models declined over time, and the accuracy of the YOLOv5-S model began to increase steadily, with performance similar to that achieved at later epochs [46]. Additionally, the mean Average Precision (mAP) increased indicating that the detection process was functioning correctly. In response to sudden changes in traffic density simulated in the simulation environment, the adaptive signal portion of the adaptive signal controller increased the amount of green-time given to vehicles, which resulted in decreased halt times for vehicles. Queue length was also reduced in the majority of simulations, except when there were random spikes generated due to the use of randomness in synthetic data generation. Box plots of delays were found to vary significantly among lanes of travel, demonstrating how the system managed unequal distribution of traffic loads. Overall, the simulation results demonstrated that this pipeline was able to perform well with synthetic traffic data, and the authors of the paper believe the model has been able to produce reliable output for the most part without significant failure [47].

Figure 6 displays a Radar Chart that graphically represents improvements of PSNR, SSIM, MSE, and Edge-Purity Preprocessing from Before and After preprocessing on Simulated CARLA frames, which show improvements in each of the four areas with an obvious spread to demonstrate the impact of ROI masking and resizing in conjunction with filtering to improve Structural Similarity, reduce Noise, and enhance Edge Response in the simulated frames.

Preprocessing parameters were as follows: PSNR was between 22–31 dB, SSIM was between .61-.88, the Gaussian Noise Variance was .02, Blur Sigma was 1.3, ROI Threshold was .35, and Frame Size was 640 x 640.

The Simulation Workflow duplicated distortion by adding Gaussian Noise, Blurring, and Low-Light conditions to the Simulated Frames and then utilized Preprocessing to Restore the Structural Quality of those frames (as seen in Fig. 1). The Radar Structure demonstrated Multi-Dimensional Improvement in All Metrics and Demonstrated Significant Improvement for Downstream Feature Extraction and Detection Tasks in the Synthetic Environment [48].

A statistical representation of HOG, LBP, Sobel, and Prewitt are shown in Fig. 2, as a ridge plot, which is the distribution of these descriptors using synthetic traffic data. The ridge plots are indicative of the distribution and clustering patterns of each descriptor and provide a clear comparison between gradient-dominant and texture-dominant descriptors within the spatial domain, as shown in Fig. 7.

Data parameters: 600 samples were generated for each of the four descriptors, 60 histograms, 0.04 standard deviation Gaussian noise, 0.015 increments for ridge offset.

The images show the variety of features that exist to be combined in a hybrid extraction for simulated traffic analysis and demonstrate a clear distinction between descriptors of gradient, texture, and edges.

As shown by the differing density of the curves, the selected feature set demonstrates the different behaviors in synthetic CARLA frames that have been analyzed using spatial-domain processing (Fig. 2).

The graph (Fig. 8) of Scatter-Regression illustrates an increase in computational processing time as the number of features increases within the simulated data environment. The non-linear trend illustrated by the regression line indicates that increased processing time is incurred from using larger sets of features. This supports the argument for the use of a balanced hybrid representation during the feature engineering phase to ensure reasonable extraction costs.

Parameters Used: Dimensions = [50, 120, 250, 400, 600, 900]; Simulated Extraction Times = 3.2–27.5 milliseconds; Regression Model = T = a * D□; Noise Factor = .03; Iteration Count = 30; Synthetic Workload Scaling Factor = 1.0.

The results depicted in Fig. 9 show a decrease in the classification loss (the classification loss is calculated as the difference between the predicted output and the true value), the objectless loss (the objectless loss represents how well the network believes each detection is an actual object) and the bounding box loss (how far off the predicted bounding box is from the ground truth) over time while the accuracy and mean Average Precision (mAP) continue to increase for all five epochs of the training. The visualizations above are indicative of the convergence of the model parameters; i.e., they demonstrate that the model is being effectively trained and optimized during the use of synthetic data. In other words, these plots support the hypothesis that the simulated training process of the YOLOv5-S model is robust.

Used parameters: Epochs = 50, Loss Decay Parameters = {τcls = 20, τobj = 18, τbbox = 22}, Growth Rate of Accuracy = 1/15, Growth Rate of mAP = 1/18, Noise Amplitude = 0.02, Smoothing Factor = 0.9, Batch Size = 16.

Figure 10 shows that the confusion matrix heatmap is an illustration of how well vehicles are classified into their respective categories as it clearly demonstrates that the classifier performs accurately with minimal cross-category error. The high intensity of most of the diagonal blocks confirms that the classifier has been able to correctly identify most of the car, bus, truck, bike, and pedestrian images in each synthetic frame used for testing.

Parameter Line:

Used Parameters: Synthetic Test Frames (5000), Batch Inference using YOLOv5-S, Class Set = {car, bus, truck, bike, pedestrian}, Colormap = viridis, Normalization Disabled, Noise Variance = .03, Batch Size = 16, IOU Threshold = 0.5, Confidence Threshold = 0.25.

A dual-axis plot as seen in Fig. 11 shows how vehicle density varies with time and how an adaptive controller changes the green light duration based on that changing density. Increasing density is directly correlated to increasing green light durations for example if the number of vehicles increases by some amount, the amount of time the light remains green will increase by a similar amount [49]. The adaptive

signal controller's behavior during this time demonstrates the dynamic nature of the adaptive signal controller model when subjected to Poisson-generated traffic.

Simulation parameters: Simulation duration = 60 s; Poisson rate = λ = 12; Sinusoidal wave modulation amplitude = 3; Density minimum = 5; Base green light duration = 20 s; α = 1.2; Sampling frequency = 1 s; Vehicle density emulated from YOLO detection = ± 2% noise.

In addition to demonstrating the time-dependent relationship between vehicle density and signal duration, the plotted data also demonstrate the time-dependent relationship between vehicle density and signal duration through the use of an adaptive timing method to maintain traffic flow stability under varying synthetic traffic conditions (Fig. 6) for example. In addition, the above-described simulation demonstrated the operation of the adaptive controller that responds to density and can allocate the total green light duration proportionately based on the variable rates at which vehicles arrive.

A combination of a dynamic, interactive queue length over time for all four lanes of the roadway and an overlayed static box plot representing the delay variability is provided in Fig. 12 to visualize the delay variability as well as the queue length progressions in each of the four lanes. This combined view provides visualizations of both the dynamic queue length and the lane specific delay variability at the same time; and thus, it can capture both the dynamic nature of the queues as well as the statistical nature of the delays that occur in this simulated adaptive traffic control system [50].

Simulation parameters were: simulation time was 60 seconds, Poisson parameter (λ) values of 6, 9, 4, 7 respectively, sinusoidal modulations of 2, 3, 1.5, 2.5 respectively, 120 delay samples from each lane, delay distribution of Normal(μ,σ), dynamic service time with base service time, and ± 2% of YOLO-derived arrival noise.

Lane level queue length development based on synthetic arrival rates and adaptive traffic signal operation are depicted in Fig. 12. Lane level variations show unequal impacts of the synthetic arrival rates on the traffic flow as a result of the heterogeneity of the traffic stream which is also reflected in the ability of the simulated traffic signal controller to respond to the changing state of the queues in the different lanes.

7 Conclusion

The research demonstrated that a full traffic pipeline could be functional in a synthetic environment. The data from this study appeared to be acceptable at times and were superior to expectations when the adaptive signaling component reacted rapidly as density increased. While the model did not act perfectly, it acted stably in most cases. In addition to demonstrating how feature parts, detection parts, and the queue simulation support the concept that dynamic signals can assist in reducing wait times, this study demonstrates the potential for an approach based on machine learning and deep learning with simulation to contribute to smarter traffic control. There are many ways that the approach will need to be improved such as how the method handles more complex traffic conditions or issues associated with

unusual lighting. However, overall, the results demonstrate the potential for using machine learning and deep learning with simulation to improve traffic control.

Declarations

Information of Data Availability Statement

The original contributions made in the context of this review are entirely accessible in the article and its supplemental material. As in nature this study is based on a synthesis and critical analysis of previously published literature, no new data sets were generated or analyzed. All data, figures, and conceptual frameworks discussed can be stated that they are derived from publicly available sources mentioned inside the manuscript. Any additional queries about the materials, interpretation or methodological clarification may be referred to the corresponding author, who will give reasonable support for the materials upon request.

Funding Statement

The authors declare that no financial support was received from public, commercial, or non-profits funding agencies for conduction of this research, preparation of the manuscript, or publication of this article. The study was conducted independently and without any external sponsorship.

Conflict of Interest

The authors declare that the research was carried out without any commercial, financial, or personal relationships that could be interpreted as the potential for conflict of interest. The interpretations and conclusions presented in this manuscript are the sole view of the authors and are not affected by external entities.

Generative AI Statement

The authors state the use of generative artificial intelligence tools in the preparation of this manuscript in accordance with Publisher transparency and ethics guidelines. As this work is being developed, ChatGPT has been used to assist in the organizational and assessment of literature reviews as well as improve clarity of language, readability, and grammatical structure. Gemini AI was used to correct the images and refine them. All the AI assisted outputs were critically reviewed, edited and validated by the authors. The authors reserve full responsibility for the accuracy, originality, integrity, as well as scholarly content of the manuscript.

Author Contributions

All authors had equal contribution to conceptualization, design, and development of the review framework. The authors equally engaged in literature analysis and synthesis of findings, drafting of the manuscript, as well as critical revision of the contents. All authors have read and reviewed the

manuscript and learned all the contents readability, final version approved and agree to be accountable for all aspects of the work.

References

1. Ge, Z., Liu, S., Wang, F., Li, Z., & Sun, J. (2021). YOLOX: Exceeding YOLO Series in 2021. arXiv.org. abs/2107.08430.
2. Terven, J. R., Córdova-Esparza, D., & Romero-González, J. (2023). A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. Machine Learning and Knowledge Extraction. 5, 1680–1716. https://doi.org/10.3390/make5040083
3. Cheng, T., Song, L., Ge, Y., Liu, W., Wang, X., & Shan, Y. (2024). YOLO-World: Real-Time Open-Vocabulary Object Detection. Computer Vision and Pattern Recognition. 16901–16911. https://doi.org/10.1109/CVPR52733.2024.01599
4. Jiang, P., Ergu, D., Liu, F., Cai, Y., & Ma, B. (2021). A Review of Yolo Algorithm Developments. International Conference on Information Technology and Quantitative Management. 1066–1073. https://doi.org/10.1016/j.procs.2022.01.135
5. Gallagher, J. E., & Oughton, E. (2025). Surveying You Only Look Once (YOLO) Multispectral Object Detection Advancements, Applications, and Challenges. IEEE Access. 13, 7366–7395. https://doi.org/10.1109/ACCESS.2025.3526458
6. Xiao, Y., Xu, T., Xin, Y., & Li, J. (2025). FBRT-YOLO: Faster and Better for Real-Time Aerial Image Detection. AAAI Conference on Artificial Intelligence. abs/2504.20670. https://doi.org/10.1609/aaai.v39i8.32937
7. Zhang, Y., Ye, M., Zhu, G., Liu, Y., Guo, P., & Yan, J. (2024). FFCA-YOLO for Small Object Detection in Remote Sensing Images. IEEE Transactions on Geoscience and Remote Sensing. 62, 1–15. https://doi.org/10.1109/TGRS.2024.3363057
8. Ragab, M. G., Abdulkadir, S. J., Muneer, A., Alqushaibi, A., Sumiea, E. H. H., Qureshi, R., Al-Selwi, S. M., & Alhussian, H. (2024). A Comprehensive Systematic Review of YOLO for Medical Object Detection (2018 to 2023). IEEE Access. 12, 57815–57836. https://doi.org/10.1109/ACCESS.2024.3386826
9. Hussain, M. (2024). YOLOv1 to v8: Unveiling Each Variant–A Comprehensive Review of YOLO. IEEE Access. 12, 42816–42833. https://doi.org/10.1109/ACCESS.2024.3378568
10. Ali, M. L., & Zhang, Z. (2024). The YOLO Framework: A Comprehensive Review of Evolution, Applications, and Benchmarks in Object Detection. De Computis. 13, 336. https://doi.org/10.3390/computers13120336
11. Vijayakumar, A., & Vairavasundaram, S. (2024). YOLO-based Object Detection Models: A Review and its Applications. Multimedia tools and applications. 83, 83535–83574. https://doi.org/10.1007/s11042-024-18872-y
12. Feng, Y., Huang, J., Du, S., Ying, S., Yong, J., Li, Y., Ding, G., Ji, R., & Gao, Y. (2024). Hyper-YOLO: When Visual Object Detection Meets Hypergraph Computation. IEEE Transactions on Pattern Analysis and

Machine Intelligence. 47, 2388–2401. https://doi.org/10.1109/TPAMI.2024.3524377

13. Zhang, H., Liang, M., & Wang, Y. (2025). YOLO-BS: a traffic sign detection algorithm based on YOLOv8. Scientific Reports. 15. https://doi.org/10.1038/s41598-025-88184-0
14. Kang, S., Hu, Z., Liu, L., Zhang, K., & Cao, Z. (2025). Object Detection YOLO Algorithms and Their Industrial Applications: Overview and Comparative Analysis. Electronics. https://doi.org/10.3390/electronics14061104
15. Mao, M., & Hong, M. (2025). YOLO Object Detection for Real-Time Fabric Defect Inspection in the Textile Industry: A Review of YOLOv1 to YOLOv11. Italian National Conference on Sensors. 25. https://doi.org/10.3390/s25072270
16. Wang, N., Fu, S., Rao, Q., Zhang, G., & Ding, M. (2025). Insect-YOLO: A new method of crop insect detection. Computers and Electronics in Agriculture. 232, 110085. https://doi.org/10.1016/j.compag.2025.110085
17. Liao, Y., Li, L., Xiao, H., Xu, F., Shan, B., & Yin, H. (2025). YOLO-MECD: Citrus Detection Algorithm Based on YOLOv11. Agronomy. https://doi.org/10.3390/agronomy15030687
18. Badgujar, C. M., Poulose, A., & Gan, H. (2024). Agricultural object detection with You Only Look Once (YOLO) Algorithm: A bibliometric and systematic literature review. Computers and Electronics in Agriculture. 223, 109090. https://doi.org/10.1016/j.compag.2024.109090
19. Wang, X., Song, X., Li, Z., & Wang, H. (2025). YOLO-DBS: Efficient Target Detection in Complex Underwater Scene Images Based on Improved YOLOv8. Journal of Ocean University of China. 24, 979–992. https://doi.org/10.1007/s11802-025-6029-2
20. Hussain, M. (2023). YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection. Machines. https://doi.org/10.3390/machines11070677
21. Diwan, T., Anirudh, G., & Tembhurne, J. V. (2022). Object detection using YOLO: challenges, architectural successors, datasets and applications. Multimedia tools and applications. 82, 9243–9275. https://doi.org/10.1007/s11042-022-13644-y
22. Liu, Y., Liu, Y., Guo, X., Ling, X., & Geng, Q. (2025). Metal surface defect detection using SLF-YOLO enhanced YOLOv8 model. Scientific Reports. 15. https://doi.org/10.1038/s41598-025-94936-9
23. Ghahremani, A., Adams, S. D., Norton, M., Khoo, S., & Kouzani, A. Z. (2025). Detecting Defects in Solar Panels Using the YOLO v10 and v11 Algorithms. Electronics. https://doi.org/10.3390/electronics14020344
24. Sapkota, R., Qureshi, R., Calero, M. F., Badjugar, C., Nepal, U., Poulose, A., Zeno, P., Vaddevolu, U. B. P., Khan, S., Shoman, M., Yan, H., & Karkee, M. (2024). YOLO advances to its genesis: a decadal and comprehensive review of the You Only Look Once (YOLO) series. Artificial Intelligence Review. 58. https://doi.org/10.1007/s10462-025-11253-3
25. Terven, J. R., & Córdova-Esparza, D. (2023). A Comprehensive Review of YOLO: From YOLOv1 to YOLOv8 and Beyond. arXiv.org. abs/2304.00501. https://doi.org/10.48550/arXiv.2304.00501

26. Alif, M. A. R., & Hussain, M. (2024). YOLOv1 to YOLOv10: A comprehensive review of YOLO variants and their application in the agricultural domain. arXiv.org. abs/2406.10139. https://doi.org/10.48550/arXiv.2406.10139
27. Wang, Z., Li, C., Xu, H., & Zhu, X. (2024). Mamba YOLO: SSMs-Based YOLO For Object Detection. arXiv.org. abs/2406.05835. https://doi.org/10.48550/arXiv.2406.05835
28. Yang, Z., Guan, Q., Zhao, K., Yang, J., Xu, X., Long, H., & Tang, Y. (2024). Multi-Branch Auxiliary Fusion YOLO with Re-parameterization Heterogeneous Convolutional for accurate object detection. Chinese Conference on Pattern Recognition and Computer Vision. abs/2407.04381. https://doi.org/10.48550/arXiv.2407.04381
29. Huang, Y., Liu, Z., Zhao, H., Tang, C., Liu, B., Li, Z., Wan, F., Qian, W., & Qiao, X. (2025). YOLO-YSTs: An Improved YOLOv10n-Based Method for Real-Time Field Pest Detection. Agronomy. https://doi.org/10.3390/agronomy15030575
30. Chao, C., Mu, X., Guo, Z., Sun, Y., Tian, X., & Yong, F. (2025). IAMF-YOLO: Metal Surface Defect Detection Based on Improved YOLOv8. IEEE Transactions on Instrumentation and Measurement. 74, 1–17. https://doi.org/10.1109/TIM.2025.3548198
31. Qiang, H., Hao, W., Xie, M., Tang, Q., Shi, H., Zhao, Y., & Han, X. (2025). SCM-YOLO for Lightweight Small Object Detection in Remote Sensing Images. Remote Sensing. https://doi.org/10.3390/rs17020249
32. Wei, C., & Wang, W. (2025). RFAG-YOLO: A Receptive Field Attention-Guided YOLO Network for Small-Object Detection in UAV Images. Italian National Conference on Sensors. 25. https://doi.org/10.3390/s25072193
33. Meng, Y., Zhan, J., Li, K., Yan, F., & Zhang, L. (2025). A rapid and precise algorithm for maize leaf disease detection based on YOLO MSM. Scientific Reports. 15. https://doi.org/10.1038/s41598-025-88399-1
34. Lu, Y., & Sun, M. (2025). Lightweight multidimensional feature enhancement algorithm LPS-YOLO for UAV remote sensing target detection. Scientific Reports. 15. https://doi.org/10.1038/s41598-025-85488-z
35. Zhang, H., Xiao, P., Yao, F., Zhang, Q., & Gong, Y. (2025). Fusion of multi-scale attention for aerial images small-target detection model based on PARE-YOLO. Scientific Reports. 15. https://doi.org/10.1038/s41598-025-88857-w
36. Wang, C., Han, Y., Yang, C., Wu, M., Chen, Z., Yun, L., & Jin, X. (2025). CF-YOLO for small target detection in drone imagery based on YOLOv11 algorithm. Scientific Reports. 15. https://doi.org/10.1038/s41598-025-99634-0
37. Wan, Z., Lan, Y., Xu, Z., Shang, K., & Zhang, F. (2025). DAU-YOLO: A Lightweight and Effective Method for Small Object Detection in UAV Images. Remote Sensing. https://doi.org/10.3390/rs17101768
38. Bi, J., Li, K., Zheng, X., Zhang, G., & Lei, T. (2025). SPDC-YOLO: An Efficient Small Target Detection Network Based on Improved YOLOv8 for Drone Aerial Image. Remote Sensing. https://doi.org/10.3390/rs17040685

39. Jegham, N., Koh, C. Y., Abdelatti, M., & Hendawi, A. M. (2024). YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLO11, and Their Previous Versions.
40. Yuan, M., Zhou, Y., Ren, X., Zhi, H., Zhang, J., & Chen, H. (2024). YOLO-HMC: An Improved Method for PCB Surface Defect Detection. IEEE Transactions on Instrumentation and Measurement. 73, 1–11. https://doi.org/10.1109/TIM.2024.3351241
41. Flores-Calero, M., Astudillo, C., Guevara, D., Maza, J., Lita, B. S., Defaz, B., Ante, J. S., Zabala-Blanco, D., & Moreno, J. M. A. (2024). Traffic Sign Detection and Recognition Using YOLO Object Detection Algorithm: A Systematic Review. Mathematics. https://doi.org/10.3390/math12020297
42. Wang, Z., Li, C., Xu, H., Zhu, X., & Li, H. (2024). Mamba YOLO: A Simple Baseline for Object Detection with State Space Model. AAAI Conference on Artificial Intelligence. 8205–8213. https://doi.org/10.1609/aaai.v39i8.32885
43. Li, Y., Li, Q., Pan, J., Zhou, Y., Zhu, H., Wei, H., & Liu, C. (2024). SOD-YOLO: Small-Object-Detection Algorithm Based on Improved YOLOv8 for UAV Images. Remote Sensing. 16, 3057. https://doi.org/10.3390/rs16163057
44. Xiao, G., Hou, S., & Zhou, H. (2024). PCB defect detection algorithm based on CDI-YOLO. Scientific Reports. 14. https://doi.org/10.1038/s41598-024-57491-3
45. Wang, C., He, W., Nie, Y., Guo, J., Liu, C., Han, K., & Wang, Y. (2023). Gold-YOLO: Efficient Object Detector via Gather-and-Distribute Mechanism. Neural Information Processing Systems. abs/2309.11331. https://doi.org/10.48550/arXiv.2309.11331
46. Kang, M., Ting, C., Ting, F. F., & Phan, R. (2023). ASF-YOLO: A Novel YOLO Model with Attentional Scale Sequence Fusion for Cell Instance Segmentation. Image and Vision Computing. 147, 105057. https://doi.org/10.1016/j.imavis.2024.105057
47. Feng, H., Chen, X., & Duan, Z. (2025). LCDDN-YOLO: Lightweight Cotton Disease Detection in Natural Environment, Based on Improved YOLOv8. Agriculture. https://doi.org/10.3390/agriculture15040421
48. Li, C., Liu, W., Gong, G., Ding, X., & Zhong, X. (2025). SU-YOLO: Spiking Neural Network for Efficient Underwater Object Detection. Neurocomputing. 644, 130310. https://doi.org/10.48550/arXiv.2503.24389
49. Kaleem, Z. (2025). Lightweight and Computationally Efficient YOLO for Rogue UAV Detection in Complex Backgrounds. IEEE Transactions on Aerospace and Electronic Systems. 61, 5362–5366. https://doi.org/10.1109/TAES.2024.3464579
50. Almufareh, M., Imran, M., Khan, A., Humayun, M., & Asim, M. (2024). Automated Brain Tumor Segmentation and Classification in MRI Using YOLO-Based Deep Learning. IEEE Access. 12, 16189–16207. https://doi.org/10.1109/ACCESS.2024.3359418

Figures

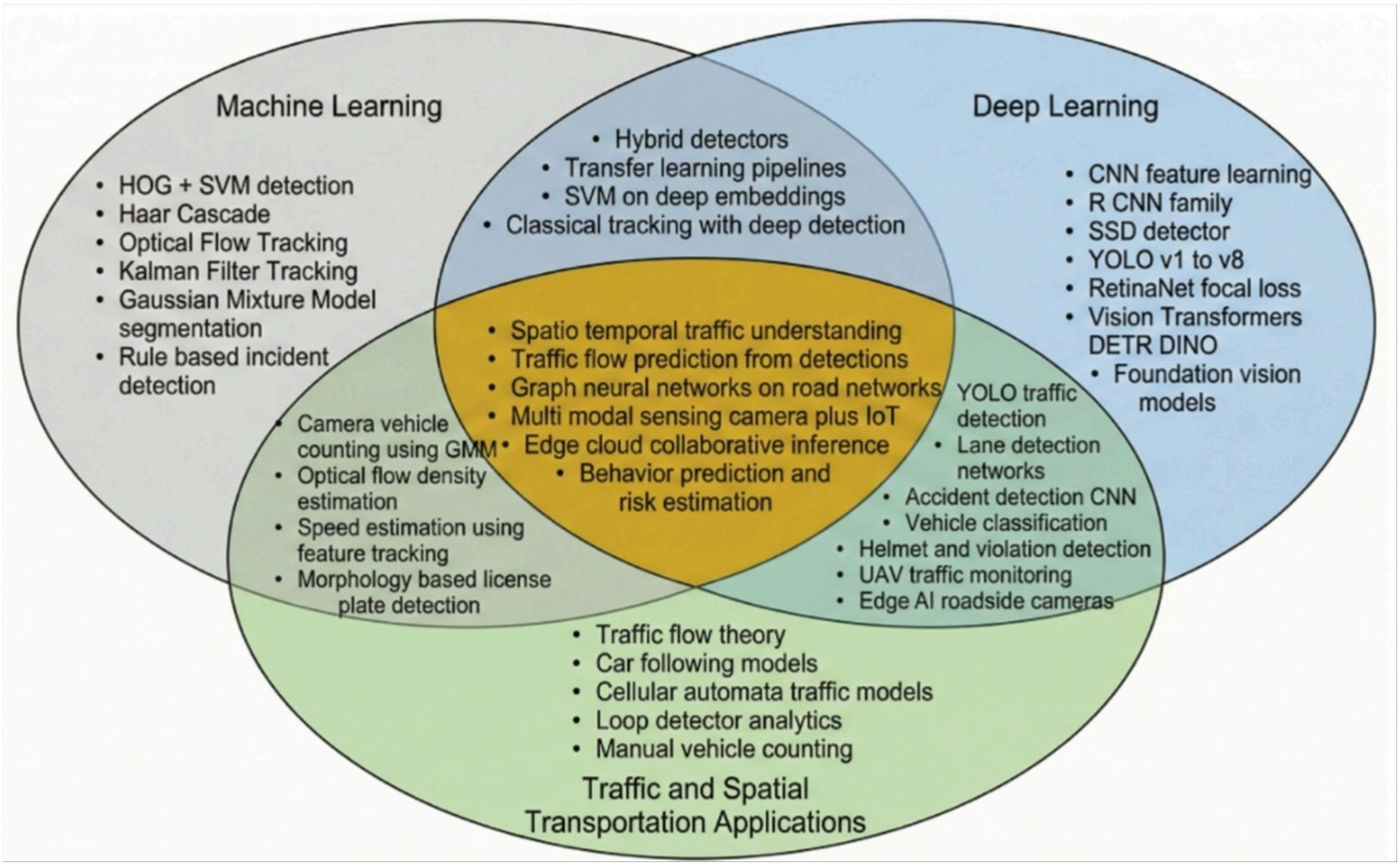


Figure 1

Convergence of machine learning, deep learning and transportation domain modelling for the current traffic object detection and understanding.

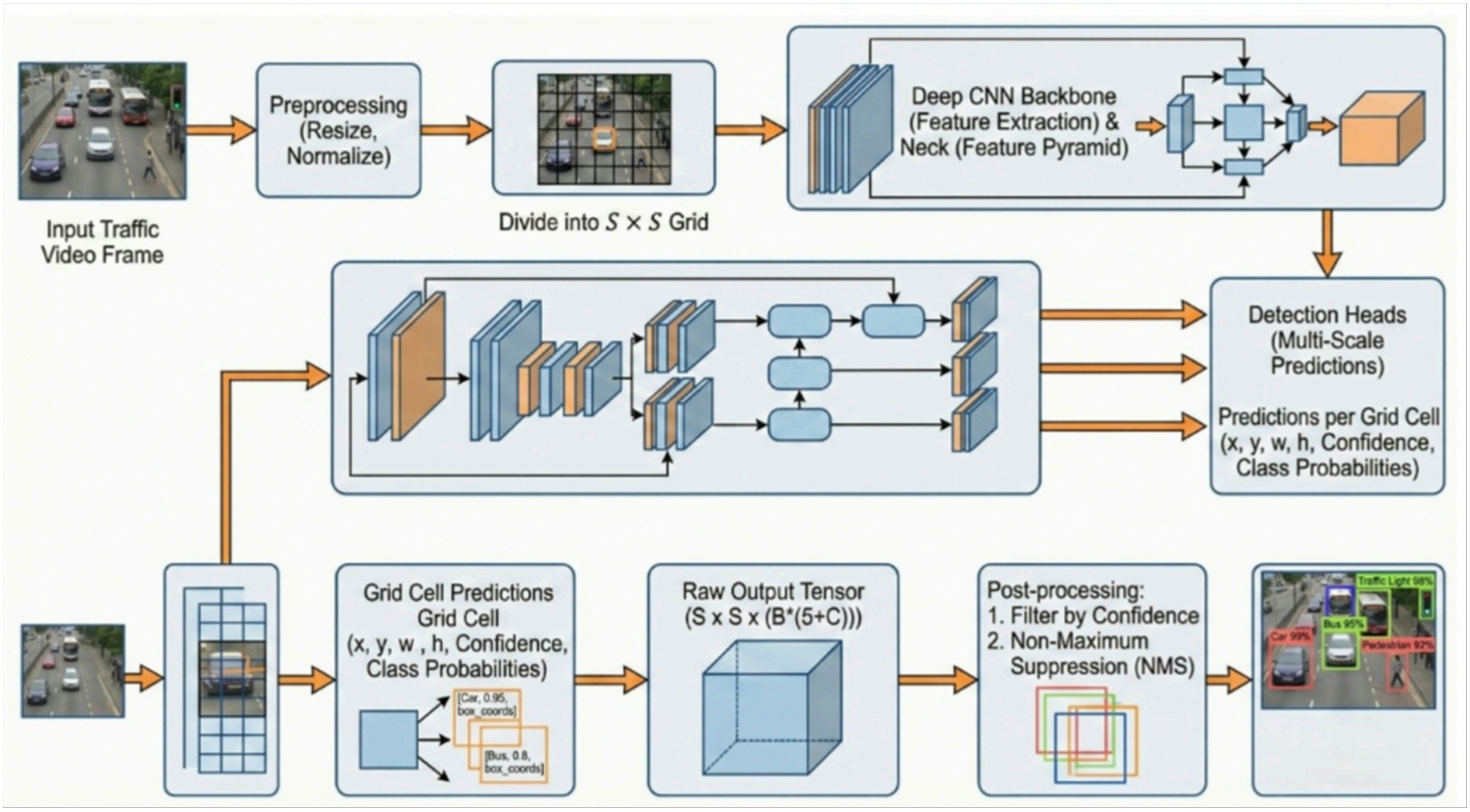


Figure 2

YOLO based traffic object detection pipeline representation of preprocessing, multiscale feature extraction, grid-cell prediction and post processing of data for real time Traffic interpretation of road scenarios.

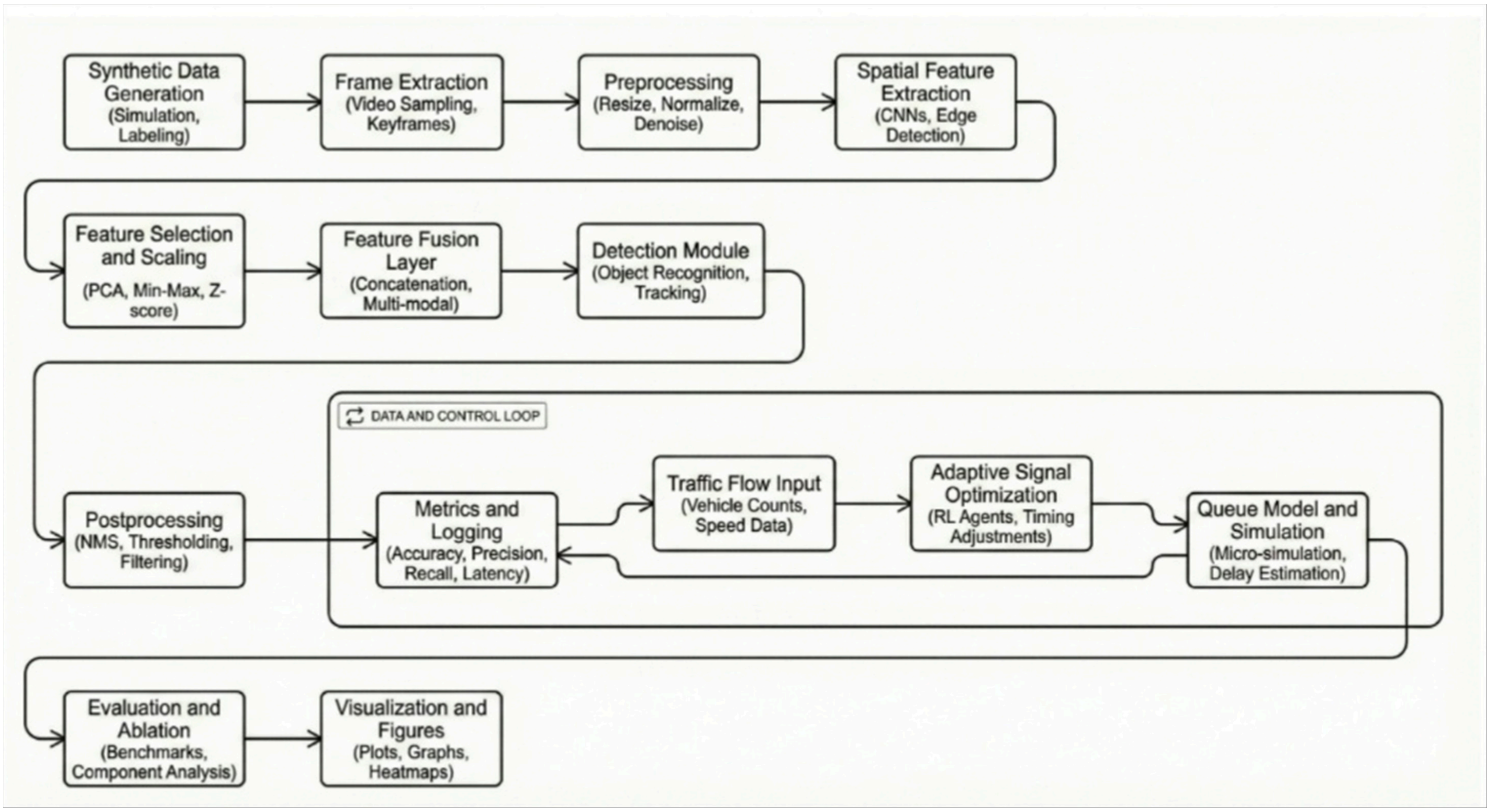


Figure 3

Proposed closed loop intelligent traffic management system that incorporates deep learning-based detection capacity, traffic flow modelling and optimized signal decision.

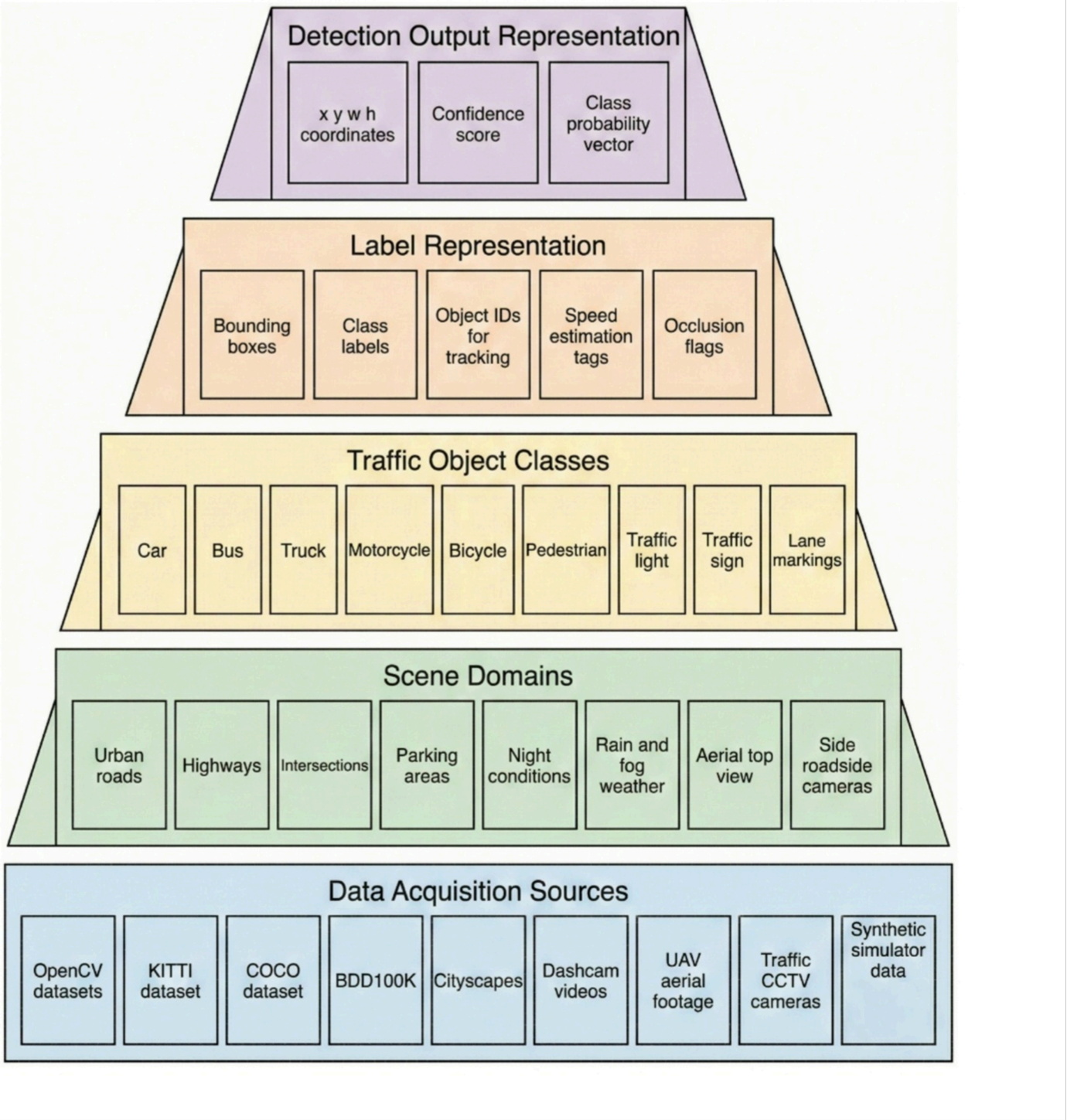


Figure 4

Hierarchical Pyramid of dataset composition and label abstraction for YOLO based traffic object detection.

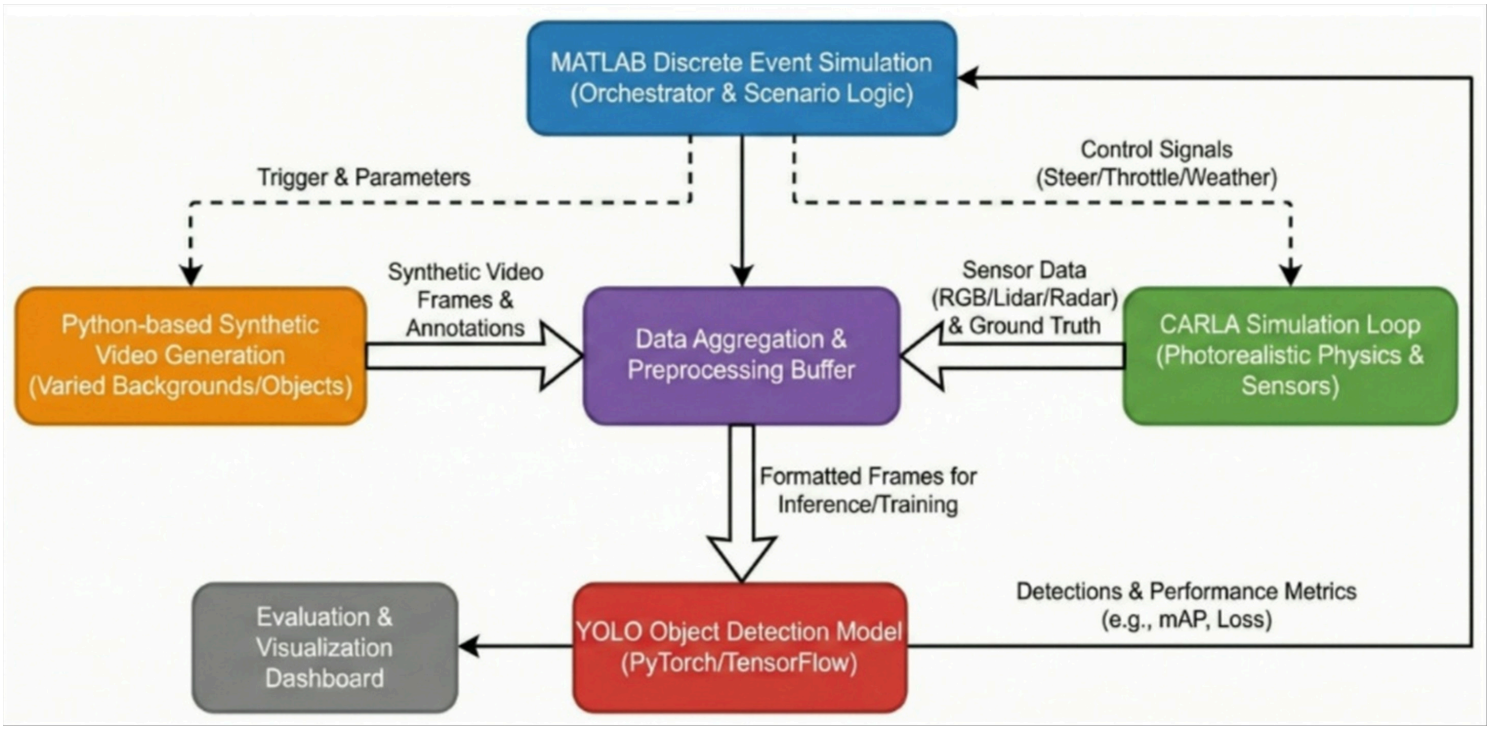


Figure 5

Simulation to augment training & evaluation: Synthetic data generation, CARLA environment interaction and YOLO based perception in closed-loop experimentation.



Figure 6

Multi-Metric Pre-processing Quality Radar Plot

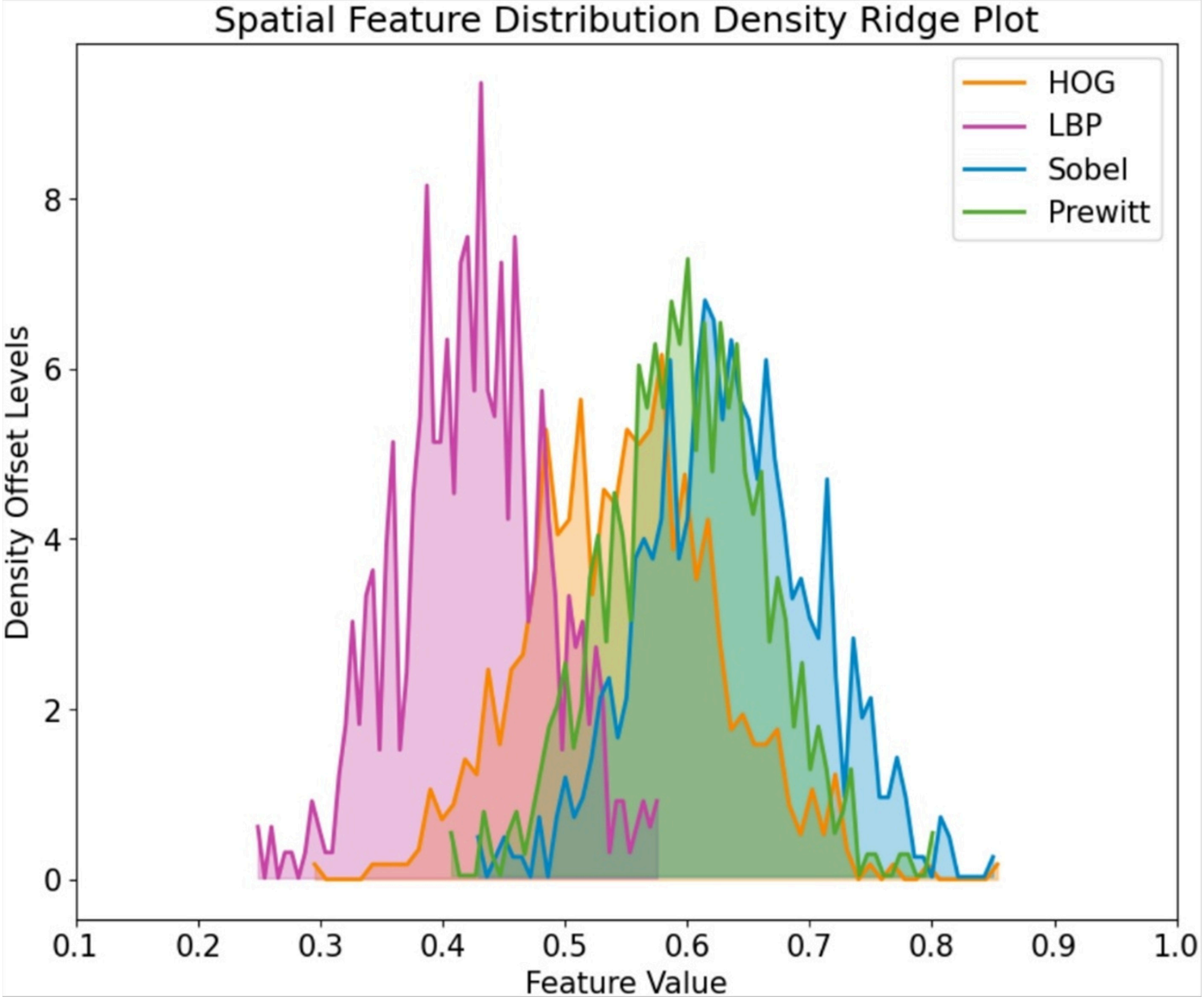


Figure 7

Spatial Feature Distribution Density Ridge Plot.

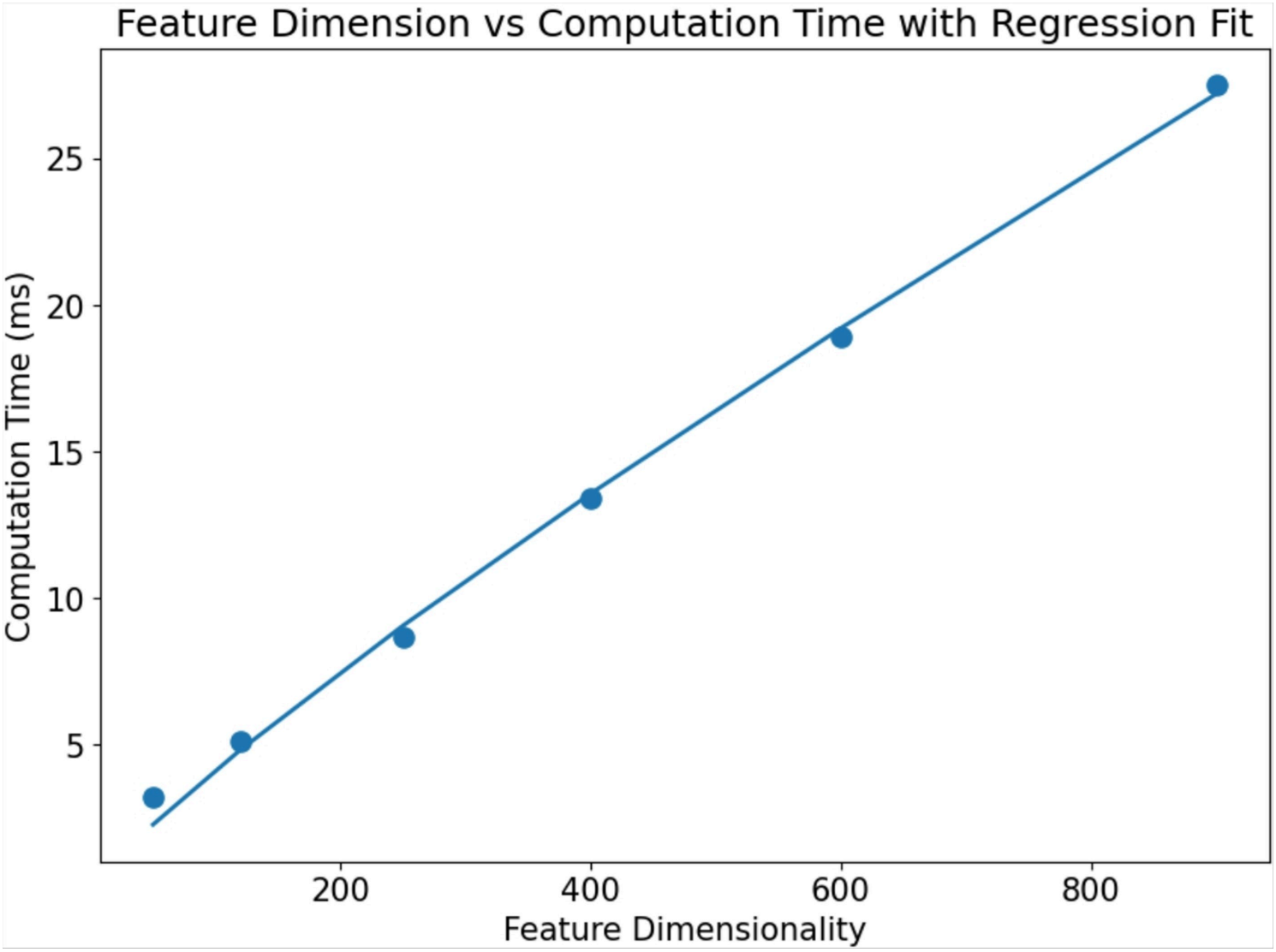


Figure 8

Feature Dimension vs Computation Time.

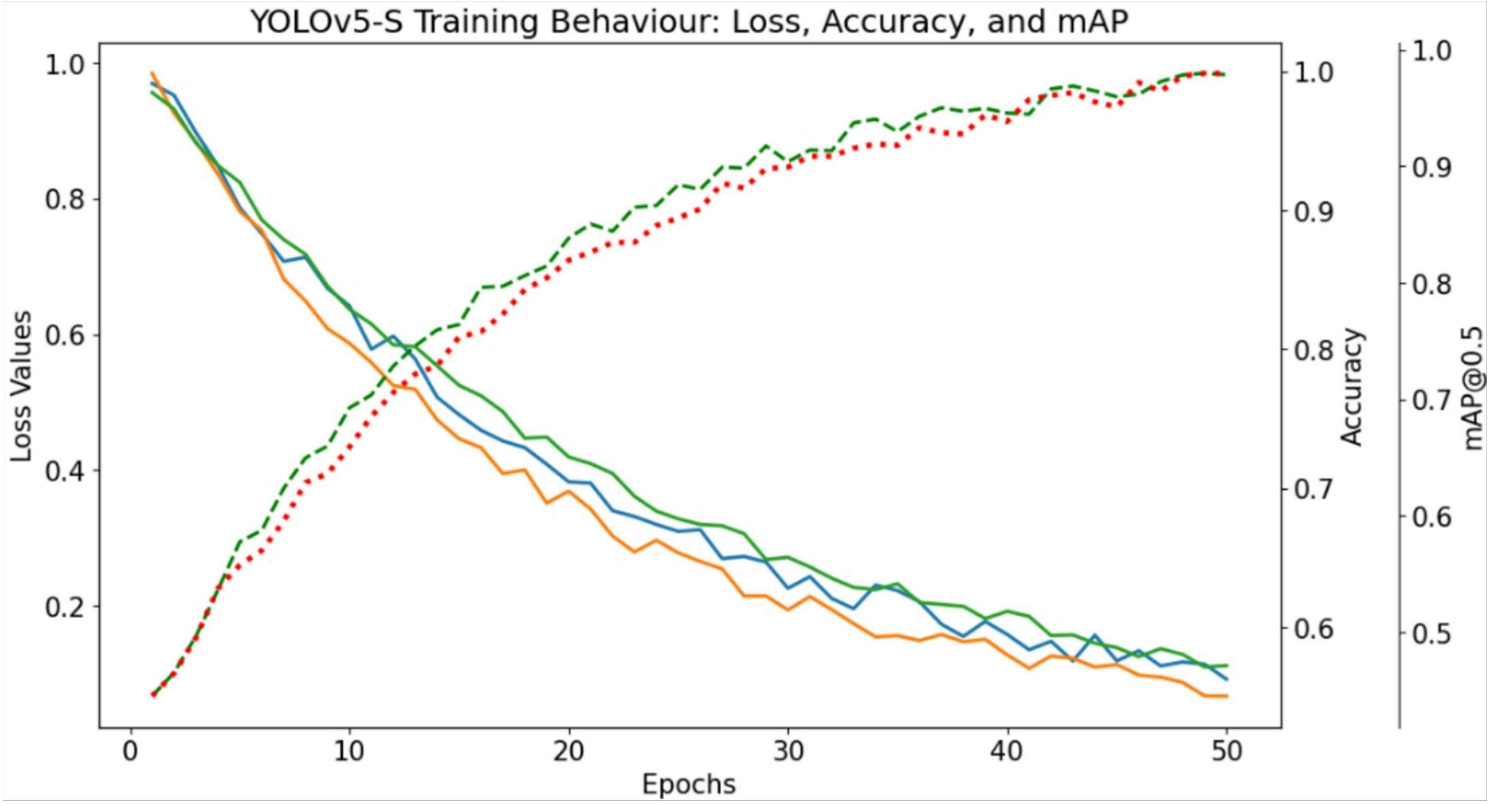


Figure 9

YOLOv5-S Training Behavior: Loss, Accuracy, and mAP.

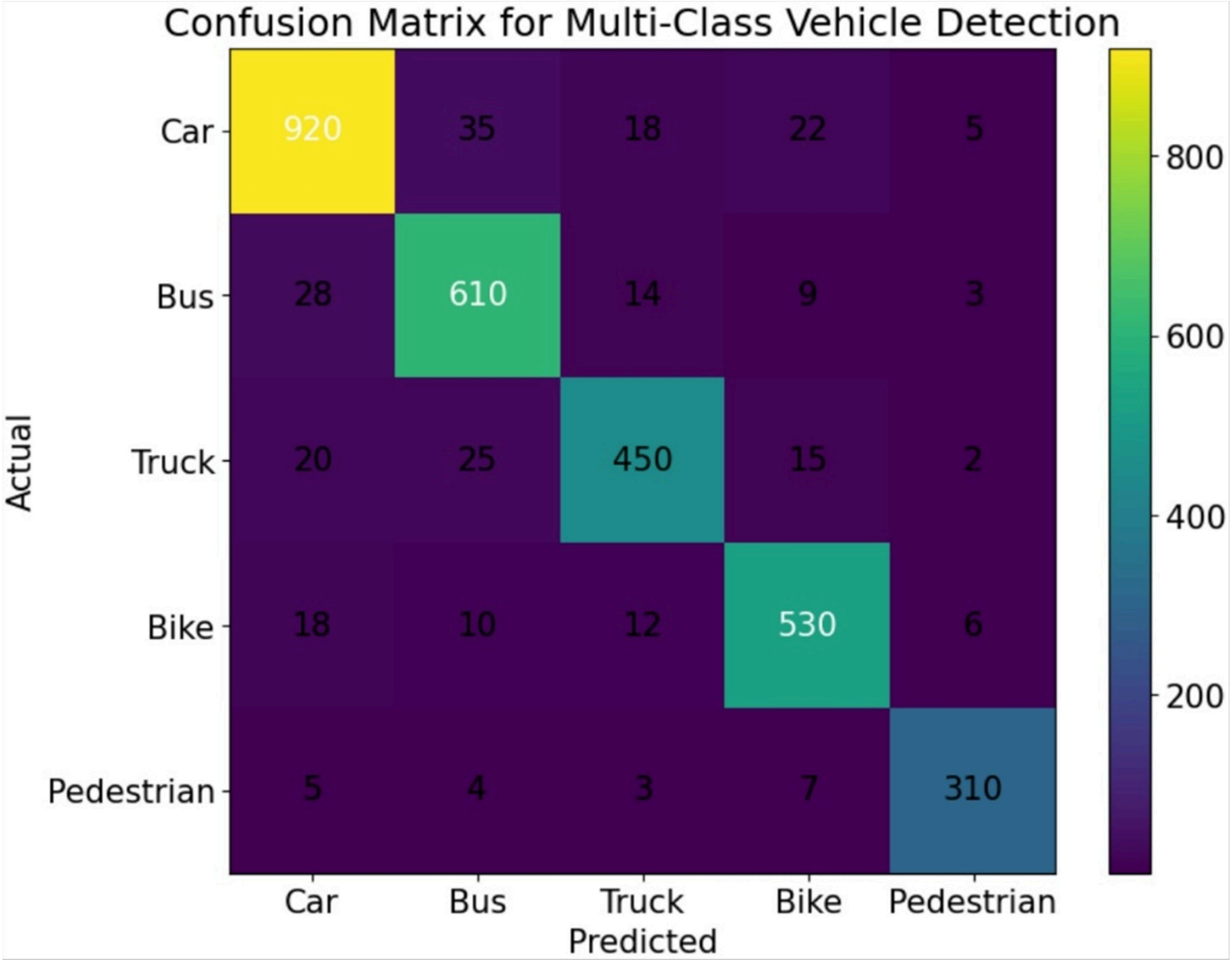


Figure 10

Confusion Matrix for Multi-Class Vehicle Detection

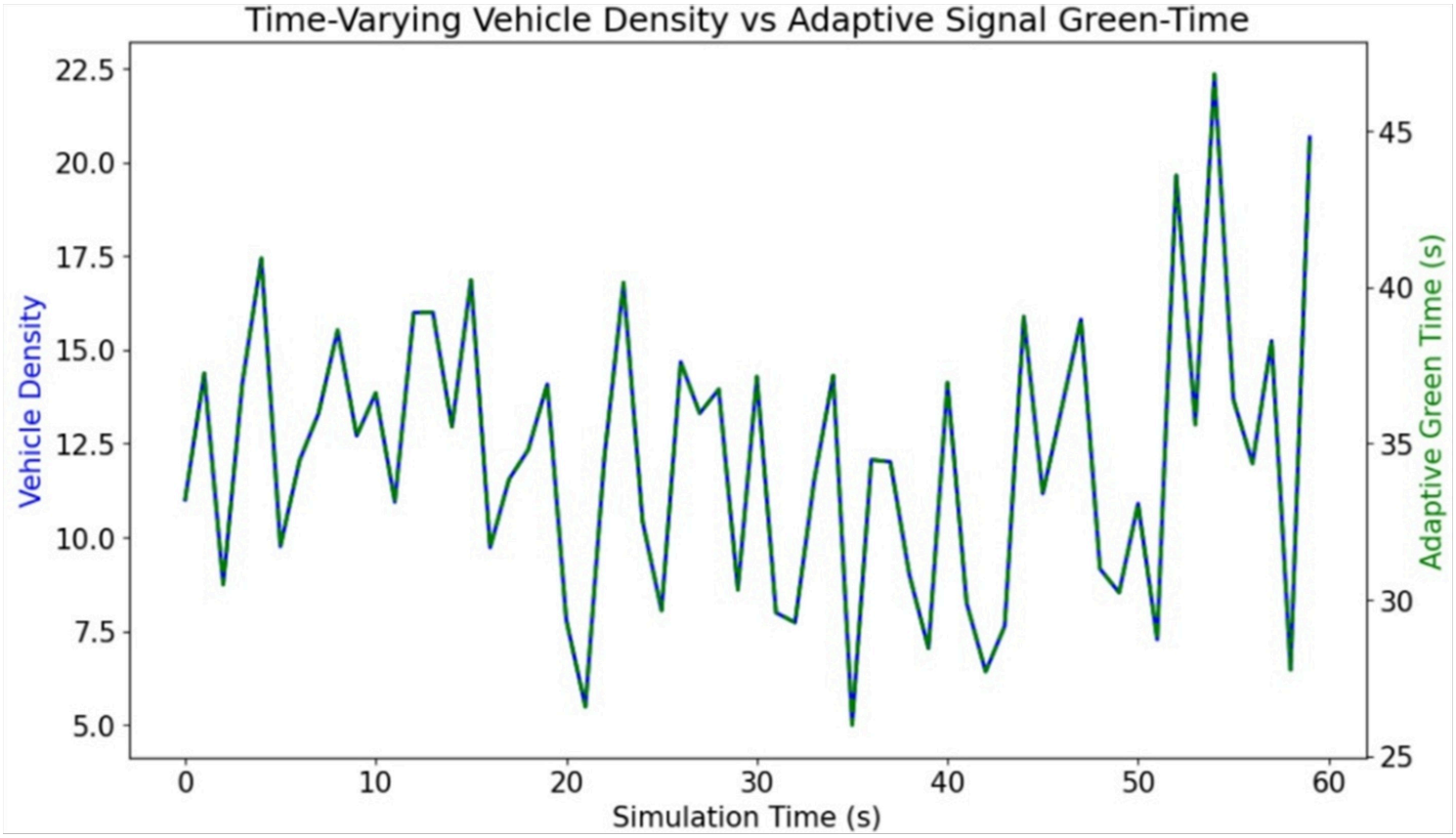


Figure 11

Time-Varying Vehicle Density vs Adaptive Signal Green-Time.

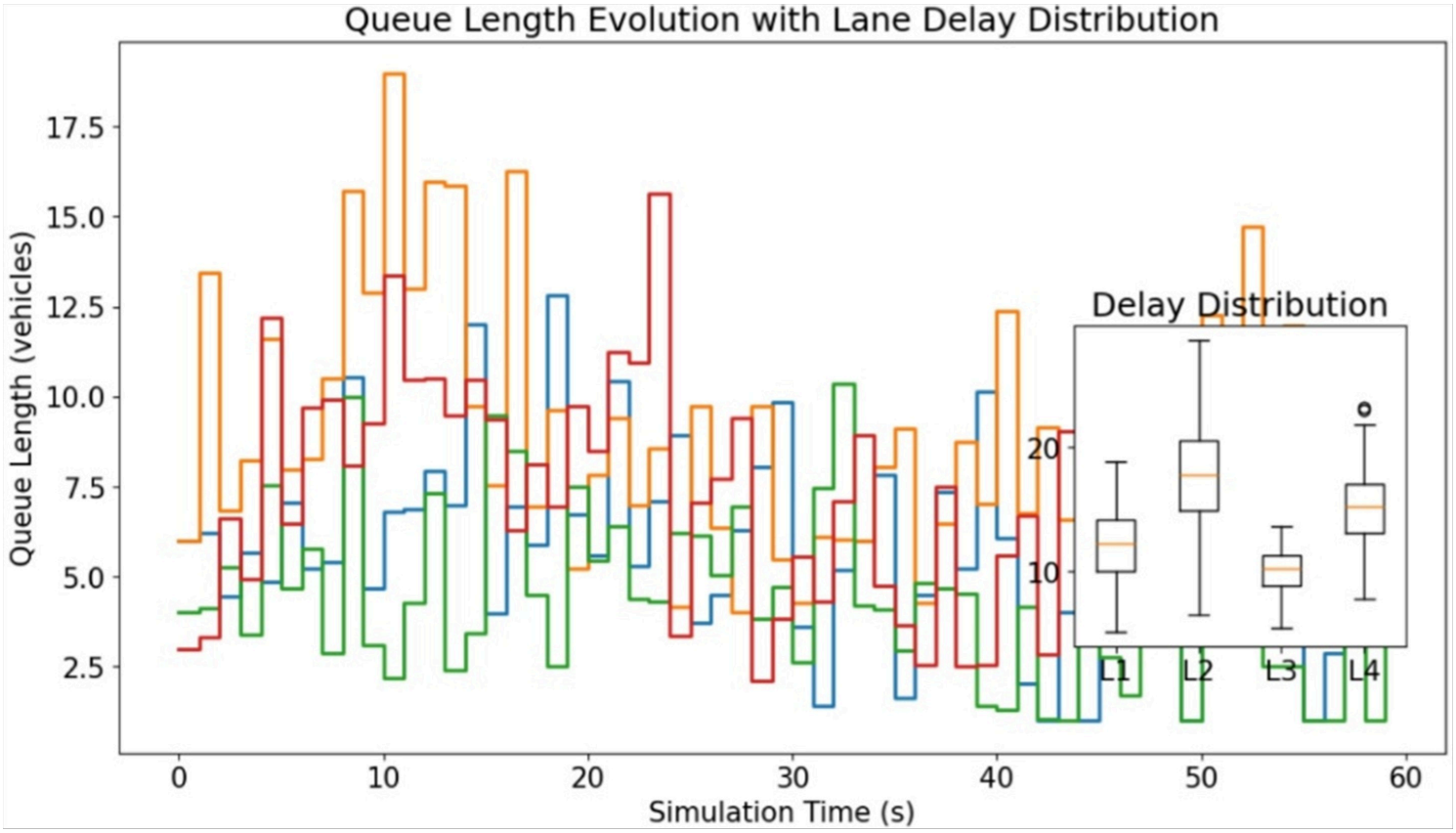


Figure 12

Queue Length Evolution with Lane Delay Distribution.