Ain Shams Engineering Journal 17 (2026) 104046

Contents lists available at ScienceDirect

Ain Shams Engineering Journal

journal homepage: www.sciencedirect.com

Full length article

Enhancing smart city transportation networks: A novel particle swarm optimization and YOLO7x-based ensemble model for accurate vehicle detection

Shahzeb Iqbal [a], Noureen Zafar [a,*], Saif Ur Rehman [a], Shireen Zafar [b], Khalid Mahmood [c], Luis Alonso Dzul Lopez [d,e,f], Eduardo Garcia Villena [d,g,h], Imran Ashraf [i,*]

[a] University Institute of Information Technology, PMAS-Arid Agriculture University Rawalpindi, 46000, Punjab, Pakistan
[b] Department of Mathematics, Rawalpindi Women University, Rawalpindi, 46000, Pakistan
[c] Institute of Computational Intelligence, Faculty of Computing, Gomal University, D.I.Khan, 29220, Pakistan
[d] Universidad Europea del Atlántico, Isabel Torres 21, Santander, 39011, Spain
[e] Universidad Internacional Iberoamericana, Campeche, 24560, Mexico
[f] Universidad de La Romana, La Romana, Republica Dominicana
[g] Universidad Internacional Iberoamericana Arecibo, Puerto Rico 00613, USA
[h] Universidade Internacional do Cuanza, Cuito, Bie, Angola
[i] Information and Communication Engineering, Yeungnam University, Gyeongsan 38541, Korea

ARTICLE INFO

Keywords:
Intelligent transportation
Traffic detection
Deep learning
Particle swarm optimization
YOLOV7

ABSTRACT

This research incorporates an intelligent transportation system (ITS) of smart cities with a newly developed ensemble model, termed EPYolov7x, through the association of particle swarm optimization (PSO) and YOLOv7x for high-accuracy traffic congestion prediction. In the proposed model, PSO is used for hyperparameter optimization for the YOLO model. Experimental results suggest a mean average precision (mAP) of 98.6% with an inference speed of 106 FPS on the UA-DETRAC dataset. It also provides good precision and real-time performance; both are essential for the practical deployment of ITS. The study findings confirm that PSO-based hyperparameter optimization improves the detection accuracy for complex traffic scenarios. The obtained results indicate superior performance compared to existing approaches. The proposed model can facilitate sustainable urban mobility toward a safer future with optimal computation.

1. Introduction

The sustainability phase calls for the development of smart cities due to global urbanization for resource production. One of the key drivers of this evolution touches upon intelligent transportation system (ITS) that play a key role in improving urban mobility, safety, and ecology [1,2]. However, congested systems present another challenge that has a huge impact not just on gross domestic product (GDP) but also on energy consumption and air pollution levels [3]. As a result, reliable traffic information requires effective and proactive traffic management coupled with effective real-time traffic forecasting. The vehicle detection and visual traffic monitoring systems have incorporated the core technology, which is a deep learning (DL) implementation, especially the series of YOLO (You Only Look Once), including YOLOv3, YOLOv5, YOLOv7x, etc. These models strike a good balance between speed and accuracy.

A significant performance gap still exists in practical applications. Despite a lot of research progress, it is still very difficult to achieve both high accuracy and real-time inference speed in difficult cases like occlusion, changing lighting, and small object scales. The performance of these hyperparameters is greatly influenced by computationally expensive manual or grid-search methods that are less than optimal whenever they are adjusted. As a result, a model is often developed that is less than optimally suited to the specific character of the flow data, limiting its use in an extensive ITS.

The larger sustainability objectives of smart cities are inextricably related to this dilemma. According to studies on sustainable energy systems [4] and the estimation of emissions from older diesel vehicles [5],

* Corresponding authors.
Email addresses: shahzebiqbal749@gmail.com (S. Iqbal), zafar@uaar.edu.pk (N. Zafar), saif@uaar.edu.pk (S.U. Rehman), shireen.zafar09@gmail.com (S. Zafar), khalid@gu.edu.pk (K. Mahmood), luis.dzul@unini.edu.mx (L.A. Dzul Lopez), Eduardo.garcia@uneatlantico.es (E.G. Villena), imranashraf92@gmail.com (I. Ashraf).

https://doi.org/10.1016/j.asej.2026.104046
Received 27 November 2024; Received in revised form 7 January 2026; Accepted 8 February 2026
Available online 10 March 2026

Nomenclature

Abbreviations

BN Batch Normalization.
CNN Convolutional Neural Network.
CSP Cross-Stage Partial.
DL Deep Learning.
ECNN Enhanced CNN.
EPYolov7x Ensemble PSO-YOLOv7x.
FPN Feature Pyramid Network.
FPS Frames Per Second.
GEP Gross Domestic Product.
ITS Intelligent Transportation Systems.
LBP Local Binary Patterns.
mAP Mean Average Precision, a crucial indicator of object detection precision.
PAN Path Aggregation Network.
PSO Particle Swarm Optimization.
SPPCSPC Spatial Pyramid Pooling with Cross-Stage Partial Connections.
SVM Support Vector Machine.
SSD Single-Shot Detection.
UA-DETRAC University of Alberta Detection and Tracking dataset.
YAML Yet Another Markup Language.
YOLO You Only Look Once (a family of object detection models).
YOLOv3 YOLO Version 3.
YOLOv5 YOLO Version 5.
YOLOv7x YOLO Version 7 – Extra Large variant.
YOLOv7-RAR A specific variant or version of YOLOv7 (e.g., possibly with "RAR" modifications).

inefficient traffic flow directly contributes to increased energy consumption and particulate matter emissions. In the same way that cutting-edge optimization algorithms, such as the Salp Swarm algorithm, are being used to boost power quality in industrial settings [6,7], complex optimization methods are also required to improve the fundamental algorithms that drive ITS. Achieving the environmental and energy efficiency goals of sustainable urban development requires superior traffic detection, which goes beyond simple computer vision tasks.

The process for extracting image characteristics has evolved as a result of developments in DL. In contrast to conventional manual feature extraction, the DL-based methods can autonomously extract features as proposed by [8]. In [9], the authors found probable sampling locations for network training input using a selective search method with DL. The problem of non-overlapping cameras detecting cars was solved by using the high-resolution vehicle-rear dataset to train a two-stream convolutional neural network (CNN). Conventional vehicle target identification requires the manual selection of appropriate functions for different settings. Convolutional functions have successfully replaced the current manual routines and produced positive outcomes.

The authors in [10] propose a traffic monitoring system that uses advanced object detection and tracking in a single pipeline. It can detect dangerous driving violations, such as wrong-way driving. The system shifts away from traditional methods, using a CNN for high accuracy in general detection (88.43%) and the ability to track illegal maneuvers (90.45%). This promotes road safety and analyzes congestion at intersections.

Along the same lines, Nocua et al. [11] focuses on motorcycle detection and tracking of real-time data using DL on embedded hardware. High accident rates are evaluated using multiple CNN models, and finally, MobileNet-v1-SSD is selected because of its high precision (90%), recall (66%), and low latency (*approx*10 ms) on NVIDIA Jetson Xavier NX. Alongside a tracker based on optical flow, this sensor can follow motorcycles and estimate their direction and speed through camera calibration. On platforms with limited resources, this work demonstrates an efficient and practical solution to monitor the traffic.

In [12], the authors combine two prediction algorithms that emphasize the importance of accurate traffic density prediction for efficient traffic control. It offers a hybrid model that is trained with annotated and augmented data and incorporates Faster R-CNN and YOLO to achieve high detection accuracy and improved traffic density prediction. The recommended approach could help the systems that regulate traffic on the roads. YOLO and Faster R-CNN both provide accuracy rates of 95.8%. The study omits meteorological data, which might have increased the accuracy and applicability of the traffic density estimations.

The authors propose the method, namely enhanced CNN with support vector machine (ECNN-SVM) based vehicle detection in [13]. To integrate feature values and identify the ideal feature set, the study uses the bat optimization approach. It uses ECNN to reduce interference from nearby moving objects and vehicles and blends SVM with local binary patterns (LBP) for bounding box creation. The testing results suggest a 96.63% accuracy rate, which is encouraging. However, this model requires a lot of processing power.

The study [14] introduces a clustering-based traffic flow prediction approach that takes into account the dynamic spatiotemporal correlations among traffic flows within a road network. This method entails dividing historical traffic data into clusters based on the similarity of spatiotemporal correlation patterns and conducting correlation analysis and predictor selection within each cluster. Multiple prediction models are individually trained for each cluster, and the model corresponding to the current traffic pattern's cluster is chosen to provide the prediction. Experimental results using real traffic data illustrate that this approach achieves strong prediction accuracy by effectively capturing the diversity in spatiotemporal correlations among traffic flows. However, it is important to note that the method assumes the spatiotemporal correlations among traffic flows to be stationary in both time and space, which may not always hold true in real-world scenarios.

In another study [15], the suggested method known as DP-SSD makes use of several feature extractors to improve prediction accuracy. Deconvolution and pooling techniques are used at various levels of the feature pyramids, improving these feature extractors. Additionally, by changing the default box's size to include smaller default boxes for DP-SSD training, the default box's range is broadened. The usefulness of DP-SSD for real-world traffic surveillance is demonstrated by experimental assessments performed on the UA-DETRAC and KITTI datasets in terms of successful vehicle detection in real-time. In particular, utilizing an input size of 300×300 and the UA-DETRAC train validation set, DP-SSD achieves a mean average precision (mAP) of 75.43%. DP-SSD achieves a mAP of 77.94 In [16], the authors proposed a YOLOv3 algorithm for road traffic prediction. The suggested strategy consists of several parts. First, a large amount of traffic data is used to train a vehicle identification algorithm based on the YOLOv3. To guarantee the model's performance on cutting-edge hardware, it is trimmed. Next, the feature extractor is retrained to optimize the DeepSORT method, which enables multi-object vehicle tracking. The multiple-object tracking network and the vehicle detection network are placed on the edge device Jetson TX2 platform. Testing is performed to validate the precision and efficacy of the proposed framework. The findings reveal that the model has an average processing capability of 37.9 frames per second (FPS)

and an accuracy rate of 92% for properly identifying traffic flow on edge devices.

The study [17] assesses how economic conditions influenced traffic congestion and private vehicle accessibility in Madrid during the 2008 economic crisis and its aftermath. It employs TomTom Speed Profiles data and Twitter data for analysis. The study reveals that as the economy recovered, traffic increased, resulting in greater congestion. Moreover, congestion patterns varied across different city areas. Notably, low-income communities in the more heavily affected southern districts encountered substantial delays during peak hours. However, it is important to note that the study does not delve into the potential influence of other factors, such as public transportation or urban planning, on traffic congestion and accessibility.

Based on the aforementioned findings, convolutional algorithms completely eliminate the conventional process of human feature extraction. Among these algorithms, there is a two-stage detection approach that utilizes the selective search area selection method. This particular algorithm is inspired by the R-CNN algorithm. In the first stage, approximately 2000 areas are selected using a selective search for each image. Since the network structure only accepts inputs of the same size, each input must be resized to 227×227. Furthermore, CNN is employed to extract features for each input, and an SVM classifier is trained to assign a suitable category score.

The study conducted by Xu et al. explores and presents these ideas [18]. The candidate boxes are adjusted using bounding box regression after some redundant candidate boxes are removed using non-maximum suppression. The two-stage algorithm's detection work is then completed. Although the two-stage target detection technique rarely runs faster than 30 FPS and requires a large amount of storage space, it is primarily distinguished by good detection accuracy as mentioned in the study by Shi et al. [19]. It was compared against the advanced performance of the object recognition algorithm in terms of generic object detection, and overall performance was assessed in terms of vehicle detection. The study by Wang et al. [20] enhanced the Faster R-CNN algorithm. This increased detection speed to 9–13 FPS and improved identification accuracy by 9.5% over the previous network. However, this study has lower accuracy compared to other advanced DL models for traffic prediction. One possible reason for this is that the study did not incorporate weather data, which is known to have a significant impact on traffic patterns. Integrating weather information into traffic prediction models can improve their accuracy by accounting for weather-related factors.

End-to-end detection is the basis of another method. The main idea is to use object detection boxes and bounding boxes to convert an object classification task into a regression problem. The algorithm eliminates the need for candidate block selection and the calculation steps involved in a two-step method by directly detecting the desired outcome. By omitting the difficult phases and simplifying the overall strategy, this concept streamlines the process. Thus, this approach is significantly faster than the two-step detection method by Khalili et al. [21]. The YOLO and the SSD algorithms are examples of representative algorithms. Since it is difficult to accelerate vehicle detection with a two-stage method, researchers have focused on one-stage detection methods. In this context, Sang et al. [22] conducted research and developed an improved version of the YOLO algorithm, known as YOLOv2 (specifically tailored for vehicle detection). However, it does not include weather data, despite its significant impact on traffic prediction results. Weather conditions like rain, snow, or fog can greatly influence traffic flow.

In DL, many established automated methods exist for optimizing hyperparameter values including grid search, random search, and Bayesian optimization; however, this study proposes a novel integration method that is specifically designed to accommodate real-time traffic conditions. The proposed novel integration method encompasses three components beyond existing practices of using particle swarm optimization (PSO) with YOLOv7x:

i. The formulation of a domain-specific fitness function that optimally combines detection performance (mAP) and real-time inference speed (FPS) into a single performance measure, which captures the trade-offs between both methods for ITS implementation,
ii. A new constrained search space that retains the inherent hardware limitations (GPU memory and power), while offering a variety of architectural configurations that can be applied to vehicular surveillance scenarios, and
iii. The development of an ensemble guidance mechanism whereby PSO not only refines individual hyperparameters but, more importantly, provides guidance for tuning multiple interacting hyperparameters (input resolution, batch size, and learning rate schedule), thereby allowing for the optimization of overall system performance.

This work represents a departure from previous hyperparameter optimization methods toward integrated system optimization for edge-AI systems, resulting in a new optimization objective of achieving maximum computational efficiency equal to accuracy in real time.

The proper anchor box for the gathered dataset was chosen using the K-means++ method, and the YOLOv2 technique was then suggested. The precision of the algorithm was 94.78%. By enhancing the SSD algorithm, Zhang et al. [15] introduced DP-SSD, and the accuracy of the algorithm was 75.43% at a speed of 50.47 FPS. The YOLOv1 algorithm which Joseph Redmon initially made known in 2016 [23], whereas the SSD strategy was presented in 2016 by Wei Liu of ECCA. Its fundamental design incorporates the Faster-anchor RCNN technology, which is an improvement over the YOLO method. The YOLO algorithm, which is thought to be the most extensively used approach for object identification, is used to calculate the bounding boxes and confidence scores for each grid cell's element by dividing a picture into a grid. In YOLOv3, an object map is used, and anchor sizes are defined for prediction. The YOLO method is said to be the basis for the application algorithm detailed in the article, and it is mentioned that YOLOv7 has a better performance compared to earlier versions. It does, however, recognize that the detection accuracy of the program may still be improved as noted by [20].

Table 1 shows a comparison of the traffic flow detection approaches. Based on the findings from the discussed research works, we contend that using commercially available object detection models alone is inadequate. The intelligent and automatic optimization of these models for the traffic domain is the key to closing this performance gap. In order to solve this, this work suggests a unique ensemble model called EPYolov7x (Ensemble PSO-based YOLOv7x), which combines the sophisticated YOLOv7x architecture with the PSO algorithm. The novelty lies in the use of PSO to automatically choose the best hyperparameters for YOLOv7x, bypassing manual fine-tuning and achieving a better balance between computational efficiency and detection accuracy.

Keeping in view these challenges, this study proposes a novel ensemble model with the following contributions

Table 1
Comparison review of traffic flow detection.

References	Theme	Proposed approach	Accuracy
[16]	Traffic Flow Detection	Yolov3	92%
[15]	Vehicle Detection	Yolov2	94.7%
[24]	Urban Vehicle Detection	Yolov7 RAR	95%
[22]	Traffic Vehicle Detection	Yolov2	94.8%
[12]	Vehicle Detection	Faster R-CNN	95.4%
[25]	Traffic congestion Prediction	Fast R-CNN	90%
[26]	Road Traffic Prediction	Yolov5	93.1%
[27]	Road Traffic Detection	Yolov4	81%

- The development of the EPYolov7x framework, which leverages PSO to optimize YOLOv7x hyperparameters for the specific task of traffic vehicle detection.
- A comprehensive evaluation demonstrating that the proposed model performs at the cutting edge of the field on the widely used UA-DETRAC benchmark dataset, surpassing existing models including YOLOv3, YOLOv5, YOLOv7x, and YOLOv7-RAR.
- Performance is further investigated in comparison to existing approaches, particularly [28,29], and [30]. The proposed approach has been shown to yield both exceptional accuracy (98.6% mAP) and real-time performance (106 FPS). This is a practical solution superior to anything else available.

The subsequent sections of this article are structured as follows. Section 2 delves into the design and methodology employed in the experiment. The findings of the experimental comparison are examined in Section 3. Lastly, Section 4 concludes all the experimental work and its outcomes.

2. Materials and methods

2.1. Dataset description

The University at Albany Detection and Tracking (UA-DETRAC) dataset [31], a sizable benchmark created for vehicle detection and tracking in difficult real-world traffic situations, is used in this investigation. A Canon EOS 550D camera mounted on road crossing bridges, offering an elevated surveillance perspective, was used to collect the dataset at 24 separate locations in Beijing and Tianjin, China. Approximately 10 hours of footage, broken down into 140,000 frames with more than 1.21 million carefully annotated bounding boxes make up the dataset. The Pascal VOC format is used for all annotations. The dataset is licensed for use in academic research, and in order to guarantee a thorough assessment under many circumstances, we used the complete, publicly available version (2015) without any subset selection.

Table 2 offers a thorough statistical summary of the dataset. The four vehicle categories in the dataset are “car,” “bus,” “van,” and “other”. As can be seen, there is a large class disparity, with the dominant class being “car.” A rigorous testbed for detection models is provided by the data, which also includes four weather situations (’sunny’, ’overcast’, ’rainy’, and ’night’), as well as different levels of occlusion and traffic density.

The total number of annotated frames is 140,000. The dataset was officially split by the providers into a training set (~60%) and a test set (~40%), which we adhered to for experiments. Bounding box coordinates (x min, y min, x max, and y max), vehicle type, a distinct tracking ID, and metadata for occlusion level and truncation are all included in the annotations. The level of blockage is classified as:

- None: There is no obstruction.
- Partial: 1% to 50% of the bounding box is obscured.
- Over 50% occlusion is considered heavy.

The square root of the pixel area of a vehicle’s bounding box determines its scale, which is classified as Small (0–50 pixels), Medium (50–150 pixels), and Large (greater than 150 pixels).

Several significant obstacles to vehicle detection are presented by the UA-DETRAC dataset:

- Severe Occlusions: Partial and heavy occlusions are frequently caused by heavy traffic.
- Significant Scale Variation: Because of the camera viewpoint, cars appear to have very different sizes.
- Complex Backgrounds: False positives are more likely in urban settings with trees, buildings, and shadows.
- Variations in Illumination: Situations might range from clear sunshine to dim nighttime conditions with headlight glare.

Table 2
UA-DETRAC dataset statistics.

Category	Number of Instances (Training)	Number of Instances (Test)	Percentage (%)
Vehicle Class			
Car	510,241	127,560	52.7
Bus	183,312	45,828	18.9
Van	98,456	24,614	10.2
Other	98,491	24,623	10.2
Weather Condition			
Sunny	302,145	75,536	31.2
Night	294,581	73,645	30.4
Overcast	212,365	53,091	21.9
Rainy	81,409	20,352	8.4
Occlusion Level			
None	601,234	150,308	62.1
Partial (1-50%)	268,451	67,113	27.7
Heavy (>50%)	20,815	5204	2.1
Vehicle Scale (Pixel Area)			
Small (0-50)	125,621	31,405	13.0
Medium (50-150)	512,884	128,221	53.0
Large (>150)	251,995	62,999	26.0

A benchmark resource for non-commercial academic research is the UA-DETRAC dataset. The dataset used in this work fully complies with its stated license agreement.

This research relies on the 2017 release of the UA, DETRAC dataset. Although a newer version (UA, DETRAC v2.0) is available, we intentionally chose the 2017 benchmark for two main reasons. Firstly and most importantly, almost all the previous works in vehicle detection, to which we compare this work, have reported their results on the 2017 version only. That is the case for the key baselines used in this study, i.e., YOLOv3, YOLOv5 [32], YOLOv7x, YOLOv7, and RAR. This decision ensures that the EPYolov7x model is compared directly and fairly with the latest state, of, the, art results under the same evaluation conditions. Secondly, the 2017 dataset already contains various challenging real, world scenarios, such as heavy occlusions, different weather conditions, and intricate urban scenes, which are sufficient to experimentally confirm the performance improvements brought by the PSO, based hyperparameter optimization framework.

The primary contribution is the new ensemble optimization method, which is tested on this standard and widely used benchmark. The next step will be to verify the proposed framework on the latest UA, DETRAC v2.0 and other recent datasets.

2.2. Data preprocessing and augmentation

Images were all resized to be uniform in size (640×640) and normalized. To improve the model’s generalization and make it robust against different physical conditions, such as the weather, lighting, and obstructions to visibility, an extensive data augmentation method was designed, using the Ultralytics library, which consists of:

- **Mosaic Augmentation:** Compositing four training images into a single mosaic to enhance contextual learning and small object detection.
- **Photometric Distortions:** Random adjustments to brightness (±10%), contrast (±10%), hue (±0.1), and saturation (±10%).
- **Spatial Transformations:** Random horizontal flipping with a 50% probability.
- **MixUp Augmentation:** Linear interpolation between image pairs and their corresponding labels to enhance regularization.

For both (YOLOv5), (YOLOv7x) and the EPyolov7x Optimized model, augmentation methods are used for both baseline models and the optimized model. The hyper-parameters associated with the augmentation techniques used are consistent with the default settings for the Ultralytics YOLOv7 implementation [29].

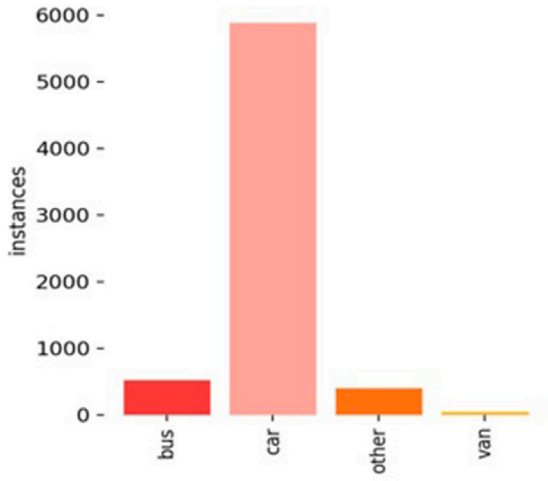

Fig. 1. Chart of labeling information for various classes.

2.2.1. Preprocessing and utilization

All frames were scaled to the 640×640 pixel input resolution of the model prior to training. Pixel values were scaled to the interval [0, 1] using conventional normalization. At this point, no additional data augmentation was used to create a baseline performance on the original dataset. The main drawback of the dataset is its substantial class imbalance and bias toward "sunny" and "night" circumstances, which the proposed model has to be able to manage with resilience. Fig. 1 (class distribution) and Fig. 2 (distribution across training/test splits and weather conditions), which are specifically cited and described in the main text, provide a visual summary of the dataset [33].

Google Colab, a cloud-based computing platform, was used for all tests. An NVIDIA GPU with 40 GB of VRAM was the main GPU resource used to produce the final results. CUDA 11.3, PyTorch 1.10.0, and Python 3.9 comprised the software environment. We were able to train the models with a batch size of 16 using this environment.

"Car", "bus", "van", and "other" are the four vehicle classifications into which the dataset is divided. The class distribution is extremely unbalanced, as Fig. 1 illustrates, with "car" being the most common category, followed by "bus", "van", and "other." Any effective model needs to be capable of managing this imbalance, which is an essential characteristic of the dataset. Furthermore, the dataset presents a variety of occlusion levels and weather conditions. Fig. 2 illustrates the overall distribution of different environmental attributes.

It includes cars, buses, vans, and other types of vehicles. Fig. 2(a) illustrates the distribution of these vehicle classes; as expected, the "car" class is the most prevalent, comprising the greatest number of instances, while the "van" and "other" classes correspond to a small percentage of the data. This skewed distribution among classes is something that the object detection model needs to take into consideration. The dataset was collected in many different real-world scenarios. Fig. 2(b) depicts the distribution of the four main weather conditions, specifically, "sunny", "night", "overcast", and "rainy". According to the chart, the dataset has a prominent representation of various lighting and weather conditions, focusing on "sunny" and "night" conditions, which are relevant for testing the robustness of the model to environmental changes.

2.2.2. Scale definition justification

We define vehicle scale as the square root of the bounding box area $\sqrt{w \times h}$, where w and h are the width and height in pixels. A transformation that converts pixel areas into evaluated linear dimensions based on perceptual sizes in the image plane. The COCO benchmark [23] takes pixel areas as thresholds (32^2, 96^2 pixels), but this study utilized a square root transformation of the respective pixel area thresholds for these reasons. First, it offers an intuitive linear metric that corresponds to object dimensions and, second, it helps normalize the distribution of scale values; therefore, threshold values (50, 150 pixels) can be selected with greater interpretability [34]. The technique also retains standard evaluation across object tracking applications while still providing a significantly more suitable approximation to a human observer's perception of object size when monitoring traffic flow conditions.

2.3. Model selection

2.3.1. YOLOv5

The YOLO algorithm by Redmon et al. [35], the first one-stage object detection algorithm is the YOLO algorithm. It abandons candidate box extraction, which exists in two-stage approaches, by converting the process of extracting the bounding box and classification into a regression problem. This is how the YOLO algorithm works. The input image is transformed into an S × S grid. Each grid is responsible for predicting

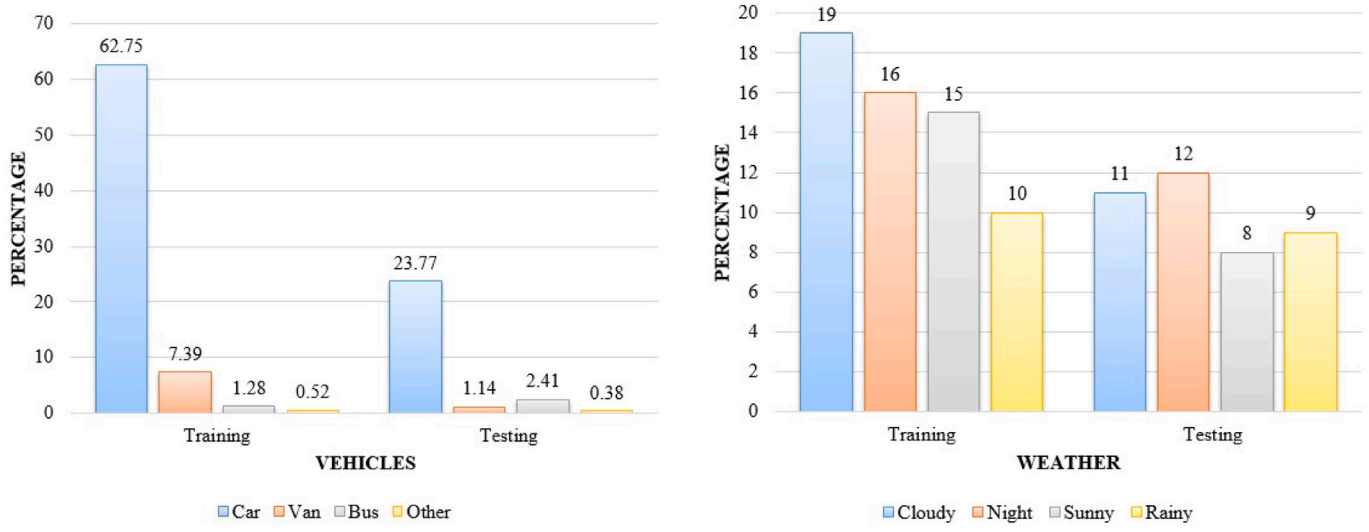

Fig. 2. The dataset overall distribution, **(a)** Types of vehicles and their percentage, and **(b)** Weather settings in the dataset.

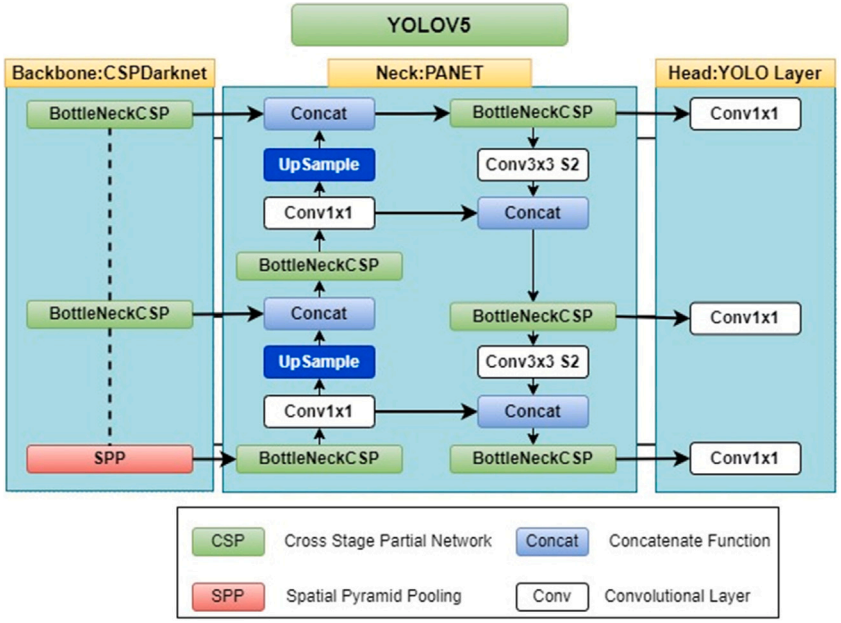

Fig. 3. Network structure of YOLOv5 [37].

the location of the target or the placement of the real box. These grids create S × S × B bounding boxes. Each bounding box is composed of five variables: the coordinates of the target's center point (x, y), the width (w) and height (h) of the target, and the confidence score, which indicates the degree to which the target is enclosed within it. Also, each S x S grid predicts the target's category probability. Each category probability is multiplied by the confidence score of a corresponding bounding box to get the category score of every predicted bounding box [15].

These forecast boxes are then processed using NMS to obtain the final prediction results. Lately, there has been rapid development in the YOLO family of algorithms. Both YOLOv4 and YOLOv5 were proposed in 2020. Within the COCO dataset, the approach of YOLOv5 balanced operational speed with an accuracy of approximately 50 mAP [36]. The YOLOv5l architecture makes use of a PANet neck combined with the CSPDarknet backbone for efficient feature extraction and multi-scale fusion. This is vital in effectively addressing the considerable scale variation of the vehicles in the UA-DETRAC dataset. Fig. 3 depicts the network structure, which includes the head, neck, and backbone.

The Focus structure and the CSP structure are two essential parts of the backbone network [38]. The YOLOv5l architecture uses a CSPDarknet backbone with Cross-Stage Partial (CSP) connections to enhance gradient flow and reduce unnecessary computations. Its neck for facilitating multiscale feature fusion is a combination of the Feature Pyramid Network (FPN) and the Path Aggregation Network (PAN). The FPN effectively transmits strong semantic features in a top-down manner, whereas the PAN ensures that precise location signals are transmitted in a bottom-up direction. This collaborative architecture is indispensable for detecting vehicles in scenes with large variations in scale. To ensure efficient spatial down-sampling, the input initially undergoes processing through a Focus module. For a detailed architectural analysis, please refer to [2], which addresses computational bottlenecks, and reduces memory usage.

2.3.2. YOLOv7x

YOLOv7 was proposed by Jian-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Switching from Leakrelu to Swish for the activation function is one of the YOLOv7 improvements. While the network still has a backbone, neck, and head, other core modules can be optimized using a residual design approach [28].

2.3.3. Backbone

Joseph Redmon created DarkNet, which serves as the YOLO algorithm's main support network. The architecture of subsequent iterations of the YOLO algorithm is optimized. The E-ELAN, CBS, SPPCSPC, and MP modules are a part of the YOLOv7 backbone network. The simplest module, CBS, integrates with other modules [28].

2.3.4. Feature fusion

Enhancing the network's capacity to learn the characteristics collected from the backbone network is the goal of the feature fusion layer. To learn as many picture qualities as is practicable, the characteristics of various granularities are first learned independently and then merged in a centralized manner.

2.3.5. Detection head

The YOLOv7 model continues to employ three detection heads, which are responsible for finding and showing information and predicted frame coordinates of the target item, along with building on the strengths of earlier algorithms. The three feature scales 20 × 20, 40 × 40, and 80 × 80 are produced by the detection heads. An individual target scale, such as a big, medium, or small target, is designated for each of the three scales [28].

YOLOv1 debuted in 2015, followed by v2 in 2016, v3 in 2018, v4 and v5 in 2020, and most recently, v6 and v7. DL has been used to produce the YOLO series [35]. YOLOv7 additionally performs case segmentation and human posture evaluation in addition to target recognition. YOLOv7 employs a single-stage approach, for instance, segmentation in the mask branch, which increases the tool's effectiveness in this area [39]. The OrienMaskHead is in use for now, but more strategies might be introduced later [40].

As seen in Fig. 4, before being processed by the backbone network, the YOLOv7 architecture makes use of a network technique in which the input picture is reduced to 640 × 640 pixels. The head layer network generates three distinct-sized feature map layers, and then the rep and convolution procedures are used to obtain prediction results. In the COCO dataset, each output (x, y, w, h, o) represents the spatial position, background data (before and after), and three anchors. There are 80 categories in the results. The output of all layers combined is therefore (80 + 5) × 3, which yields a size that is 255 times greater than the feature map.

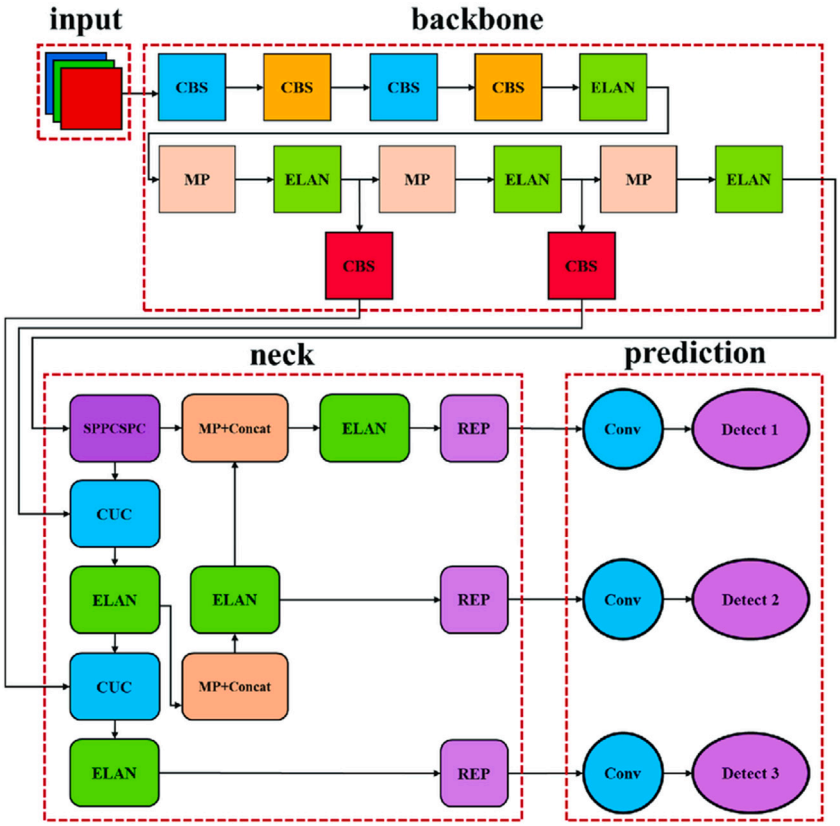

Fig. 4. Yolov7x overall framework network process [41].

To improve feature learning and multi-scale context aggregation, the YOLOv7x architecture adds sophisticated elements like the Extended-ELAN (E-ELAN) computational block and an improved SPPCSPC module. A thorough description of the YOLOv7x network mechanism and its essential components may be found in Fig. 4 [31].

The YOLOv7 architecture may be divided into an input network, a backbone network, and a head network, as shown in the structure diagram [42]. The YOLOv7 network follows a specific preprocessing and down-sampling procedure before feeding the image into the Fig. 4 backbone network. Here is a summary of the process:

i. **Preprocessing:** The input image is preprocessed using various techniques to enhance its quality or normalize it for better performance during training and inference.
ii. **Downscaling:** The preprocessed image is down-sampled to a size of 640×640 pixels and has three color channels (RGB). This down-sampling helps reduce the computational complexity of the subsequent network layers.
iii. **CBS Composite Module:** The down-sampled feature map is passed through the CBS composite module. This module applies a series of operations, including convolution, batch normalization (BN), and an activation function to the input feature map. These operations help extract and transform features from the input data.
iv. **ELAN Module:** After the CBS composite module, the feature map's width and length are halved by the ELAN module. This module employs specific techniques to reduce the spatial dimensions of the feature map while preserving important information.
v. **MP Module:** The feature map is further down-scaled by the max-pooling module. This module performs max-pooling operations by choosing the highest value in each pooling zone to decrease the feature map's spatial dimensions.
vi. **Increasing Output Channels:** To match the doubled number of input channels resulting from down-sampling, additional output channels are simultaneously added. This step ensures consistency in the number of channels throughout the network architecture.

Overall, these preprocessing and down-sampling steps in YOLOv7, along with the application of the MP module, the ELAN module, and the CBS composite module, help extract and transform features from the input image while reducing its spatial dimensions. This process is crucial for subsequent object detection and localization tasks performed by the YOLOv7 network.

Several methods, including increasing cardinality, mixing, and scrambling, were used to improve the network's learning capacities while keeping the initial gradient flow. In order to guarantee that the feature map ensembles retained the same channel count as the original design, group convolution was used to enhance the channel count and computing block cardinality. This leads us to the following point: The number of channels in the ELAN component's output is double its input. The size of the feature map and the number of layers were reduced by half by the top branches of the MP component's max-pooling operation. The bottom branch increased the batch size by one and the duration by two after the initial iteration, while halving the length and breadth of the feature map. The two tiers of the tree merged into one. After making all of these changes, an equal-sized feature map with input and output channels was produced [43].

This work used two cutting-edge object detectors, YOLOv5l (large variant) and YOLOv7x (extra-large variant), to set a performance baseline and show a progressive improvement pathway. YOLOv7x's architectural improvements, such as the Extended-ELAN (E-ELAN) computing block and an improved compound scaling technique, which are intended to provide a better accuracy-efficiency trade-off on complex datasets, were the driving force behind the switch from YOLOv5l to

Table 3
Baseline model training configurations.

Parameter	YOLOv5l	YOLOv7x
Input Image Size	640×640	640×640
Batch Size	32	16
Optimizer	SGD	SGD
Initial Learning Rate	0.01	0.01
Momentum	0.937	0.937
Weight Decay	0.0005	0.0005
Training Epochs	50	50

YOLOv7x. The PyTorch framework (v1.10.0) was used to create both models, and the UA-DETRAC dataset was used to train them from scratch. Without making any structural changes, the basic architectures were applied as specified in their original publications [28,43]. Optimizing the training settings for the vehicle detection task was the main goal. Table 3 gives the baseline models' explicit training parameters.

A balanced view of both detection accuracy and inference speed is provided by the two main metrics: mean Average Precision at an IoU threshold of 0.5 (mAP@0.5) and Frames Per Second (FPS). These metrics were computed after training was completed, on the official UA-DETRAC test set.

This investigation adopted a progressive experimental approach, starting from Yolov5l up to Yolov7x. The architectural improvement of the latter for a better trade-off between accuracy and computational efficiency on complex, real-world datasets, such as UA-DETRAC, was the reason behind this shift [28]. Both models were implemented on PyTorch.

2.4. PSO and YOLOv7x-based ensemble model

Object detection models deployed within an urban ITS necessitate very high accuracy and real-time inference capabilities. Performance for the YOLOv7x architecture is highly sensitive to its hyperparameters, e.g., initial learning rate, momentum, weight decay, and optimizer choices. Identifying the optimal configuration of this high-dimensional and intrinsically non-linear search space constitutes a crucial yet nontrivial task.

Traditional methods of hyperparameter optimization are poorly suited to this challenge. Grid search becomes computationally intractable due to its exponential time complexity relative to the number of parameters. While random search is more sample-efficient compared to the grid search, it is still a passive strategy that does not use information from previous evaluations to inform future searches. This leads to long optimization times and poor resource utilization.

PSO was adopted for this work because, with its demonstrated capability to explore complex optimization landscapes effectively, it provides several advantages in real-world ITS deployment:

i. **Sample Efficiency and Directed Convergence:** As a metaheuristic algorithm, PSO relies on a population-based search that is guided by social interaction. Each particle's movement is informed by a balance between its own historical best position, pbest, and the swarm's global best, gbest. This mechanism enables PSO to exploit promising regions of the hyperparameter space efficiently and makes it converge to a robust solution in fewer iterations compared to random search, which is of prime importance because training YOLOv7x is computationally expensive.
ii. **Real-World Viability:** The computational efficiency of PSO directly translates to reduced development time and lower costs related to cloud/GPU resources. This makes a model development lifecycle more agile and economically feasible for smart city implementations, where rapid iteration and deployment are often required.
iii. **Gradient-Free Optimization:** PSO does not require the objective function-in this case, the validation mAP-to be differentiable. Since hyperparameters essentially lack gradient information, PSO becomes a strong and flexible optimizer.
iv. **Alignment with Smart City Objectives:** This systematic capability of PSO to find a high-performance configuration ensures that the final model can operate at its fullest potential, maximizing the detection accuracy for public safety and minimizing latency for real-time traffic management, which is at the heart of core objectives such as safety, efficiency, and sustainability in urban mobility.

The integration of PSO with YOLOv7x formalizes the hyperparameter tuning process. Let a particle's position vector $x_i = (lr, momentum, decay, \ldots)$ represent a candidate set of hyperparameters for YOLOv7x. The PSO algorithm proceeds as follows:

i. **Swarm Initialization:** A population of N particles is initialized with random positions $x_i(0)$ and velocities $v_i(0)$ within predefined bounds for each hyperparameter.
ii. **Fitness Evaluation:** For each particle i at iteration t, the fitness $F(x_i(t))$ is evaluated. This involves:
   - Configuring YOLOv7x with the hyperparameters from $x_i(t)$.
   - Training the model for a reduced number of epochs (for computational efficiency).
   - Evaluating the trained model on a held-out validation dataset. The fitness function is the mAP0.5:0.95, which we seek to maximize for the best position update.
   - Update the particle's personal best: $pbest_i(t+1) = \begin{cases} x_i(t+1), & if\ F(x_i(t+1)) > F(pbest_i(t)) \\ pbest_i(t), & otherwise \end{cases}$
   - Update the swarm's global best: $gbest(t+1) = argmax_{pbest_i(t+1)}[F(pbest_i(t+1))]\ for\ i = 1, \ldots, N$
iii. **Velocity and Position Update:** The swarm is propelled toward optimum regions by updating each particle's velocity and position:

$$v_i(t+1) = \omega \cdot v_i(t) + c_1 r_1 \otimes [pbest_i(t) - x_i(t)] + c_2 r_2 \otimes [gbest(t) - x_i(t)] x_i(t+1) = x_i(t) + v_i(t+1) \quad (1)$$

where $\otimes$ denotes element-wise multiplication, ω is the inertia weight, c_1 (cognitive coefficient) and c_2 (social coefficient) control the influence of $pbest$ and $gbest$, and r_1, $r_2 \sim U(0,1)$ are vectors of random numbers introducing stochasticity.

This process iterates until a stopping criterion (e.g., max iterations, convergence tolerance) is met. The hyperparameters from the final $gbest$ are used to train the definitive EPYOLOv7x model.

The normal YOLOv7x architecture serves as the foundation for the proposed model, known as EPYOLOv7x (Ensemble PSO-tuned YOLOv7x), without any structural changes, unique layers, or extra modules. The Extended-ELAN (E-ELAN) computational blocks for effective feature learning, the SPPCSPC module for multi-scale context aggregation, and the compound scaling strategy for optimal model capacity, all architectural improvements built into YOLOv7x, were applied as described in the original work by Wang et al. [28].

The systematic hyperparameter optimization procedure utilizing Particle Swarm Optimization (PSO) is what makes the EPYOLOv7x model novel, not its architecture. The hyperparameter configuration of the typical YOLOv7x model (e.g., learning rate, batch size, optimizer settings) has a significant impact. Manual tweaking is frequently computationally costly and inefficient. In order to overcome this, we implemented a PSO-based ensemble search to quickly and automatically identify an optimal hyperparameter set designed especially for the vehicle recognition task on the UA-DETRAC dataset. This is the main change that this study suggests. This strategy is justified in three ways:

- **Efficiency:** Compared to random or grid search, PSO's directed search finds a strong solution more quickly.
- **Performance Maximization:** Without altering the basic architecture of the main YOLOv7x model, it systematically enhances the detection accuracy in the form of mAP.
- **Automation:** It provides a repeatable approach to model optimization by eliminating the need for manual incremental tuning.

Section 2.5 describes the final configuration that was adopted to train the definitive EPYOLOv7x model, including the particular hyperparameters optimized by PSO. This PSO-driven optimization is the main development differentiating EPYOLOv7x from its baseline and explaining its performance gains.

2.5. YOLOv7x architecture and strategic modifications

The YOLOv7x architecture serves as a powerful backbone, which was strategically selected and modified for the traffic detection domain:

- **Extended-ELAN (E-ELAN):** The core computational block of YOLOv7x uses an E-ELAN design. It employs grouped convolutions to expand the channel cardinality and continuously shuffles and merges grouped features. This architecture strengthens the network's feature learning capability without disrupting the original gradient path, which is crucial for maintaining stable training and achieving high accuracy on complex traffic datasets with diverse vehicle types and scales.
- **Compound Model Scaling:** The "x" variant denotes a model scaled up in depth and width. This compound scaling provides the necessary capacity to model the complex patterns and high object densities typical of urban traffic scenes, directly contributing to the model's superior accuracy.
- **Spatial Pyramid Pooling with Cross-Stage Partial Connections (SPPCSPC):** A key architectural enhancement over prior versions is the refined SPPCSPC module. This structure aggregates contextual information at multiple receptive fields, allowing the network to better understand the context around vehicles, crucial for handling occlusion and varying vehicle sizes, while the cross-stage partial (CSP) design maintains computational efficiency, thus preserving high inference speed (FPS).

An example of a typical method from the family of swarm intelligence is PSO [15]. It was first developed in 1995, and it was modeled after the way a flock of birds changes its location in search of food based on both its collective position and each individual bird's previous location. The algorithm's effectiveness and resilience have been demonstrated when applied to a variety of high-dimensional real-world applications [17–19]. It is a population-based meta-heuristic optimization approach, which means it first creates a variety of separate search "particles" that individually represent potential solutions. Then, through an evolutionary process, this population of particles modifies its placements. PSO has the advantage over other swarm intelligence algorithms in that it can quickly and effectively navigate a huge, multidimensional search space. Although it cannot guarantee that it will identify the global optimal solution, it is likely to do so in a reasonably short period of time (iterations).

Particles are first initialized with a random location P and velocity V without any prior information. Each generation updates the velocity by taking into consideration the previous velocity, the particle's personal optimum Pbest, and the population's most recent optimal solution Gbest. After that, based on the current position and velocity, the position is updated. The mathematical representation is shown in Eqs. (2) and (3). The generation is indicated by the superscript.

$$V^i = V^{(i-1)} + C1 \cdot R1 \cdot (P_{best}^{(i-1)} - P^{(i-1)}) + C2 \cdot R2 \cdot (G_{best}^{(i-1)} - P^{(i-1)}) \quad (2)$$

$$P^i = P^{(i-1)} + V^i \quad (3)$$

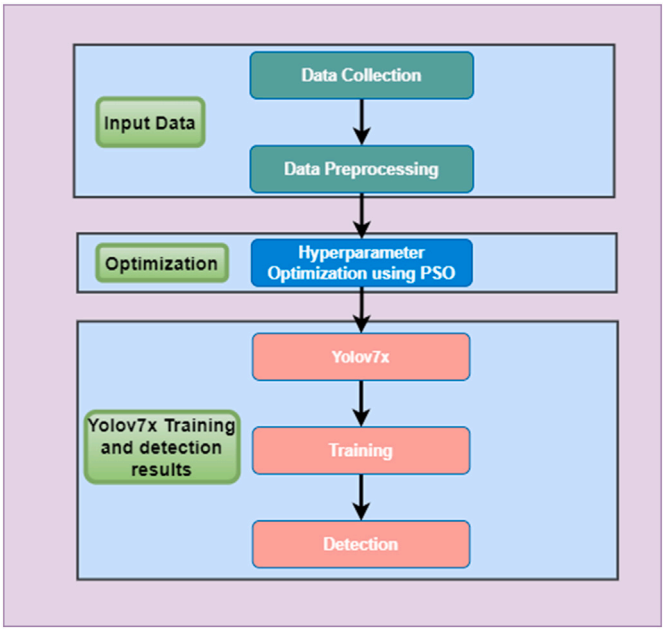

Fig. 5. YOLOv7x detection.

The terms "cognitive parameter" and "social parameter" refer to the acceleration coefficients c_1 and c_2, respectively, and "coefficient ω" stands for the inertia weight. The three coefficients are used to balance how quickly a particle learns on its own compared to how quickly the PSO population learns. r_1 and r_2 are values that are produced at random, adding a stochastic element to the search process to increase the size of the search space covered.

i. **Data Collection - UA-DETRAC:** We have collected a dataset of traffic images, specifically the UA-DETRAC dataset. This dataset contains images and annotations for various traffic scenarios. These steps are shown in Fig. 5.
ii. **Preprocessing Image Data:** Preprocessing of the traffic image data involves various steps, including resizing images to a uniform size suitable for YOLOv7x. Data augmentation techniques like random cropping, flipping, and rotation increase the dataset size and diversity. Normalization of pixel values ensures consistent input to the model.
iii. **Ensemble-Based PSO Algorithm for Hyperparameter Optimization:** An ensemble of YOLOv7x models for object detection has been designed. We used the PSO algorithm to optimize the hyperparameters of this ensemble. The hyperparameters in this case include learning rates, batch sizes, and anchor box sizes. PSO helps in finding the optimal set of hyperparameters by optimizing (i.e., either minimizing or maximizing) a chosen objective function based on validation performance metrics like mAP.
iv. **Training YOLOv7x Model:** After hyperparameter optimization, we used the tuned hyperparameters to train the YOLOv7x model on the UA-DETRAC dataset. Training involves feeding the preprocessed traffic image data into the model and iteratively adjusting model weights based on the loss function (typically a combination of localization and classification losses). The trained model learned to detect objects like vehicles in traffic images.
v. **Object Detection Results:** After training, we evaluated the YOLOv7x model's performance on a separate dataset, possibly a validation or test set from the UA-DETRAC dataset. We measured the model's accuracy, precision, recall, F1 score, and mAP

to assess its ability to detect objects in traffic images. The final detection results provide insights into how well the YOLOv7x model can identify and locate objects of interest in real-world traffic scenarios.

As a case study, we tested the algorithm on the UA-DETRAC image classification dataset using cross-entropy as an optimization performance metric. The UA-DETRAC dataset used to evaluate the algorithm is freely available and has established itself as a test and benchmark set within the community [26]. The UA-DETRAC dataset, introduced by Lyu et al. in 2017, is a valuable resource for vehicle detection tasks. It primarily comprises data captured in Beijing and Tianjin at road crossing bridges. The dataset was collected using a Canon EOS 550D camera, resulting in images in JPEG format. The dataset is categorized into four main classes: Car, Bus, Van, and Others. These classes represent different types of vehicles. Notably, the majority of the dataset falls under the ”vehicle” category, which includes cars, buses, vans, and possibly other types of vehicles. Both training and test sets are provided within the dataset, making it suitable for machine learning and evaluation purposes. The dataset has a size of 9.16 gigabytes (9.16G), indicating its substantial volume of data. Classification of this set requires a challenging level of abstraction that is achieved by a sufficiently deep network to test the algorithm on.

Optimizing hyperparameters for YOLOv7x using PSO is a complex task that involves tuning a range of parameters to improve the model's performance. Here are the steps and some hyperparameters that can be considered for optimization using PSO:

2.5.1. PSO algorithm

i. **Select Hyperparameters to Optimize:**
   - **Learning Rate (lr):** The YOLOv7x model learning rate controls how quickly the model adapts during training. It is a crucial hyperparameter to tune.
   - **Batch Size:** The batch size determines how many samples are processed in each training iteration. Finding an optimal batch size can impact training efficiency.
   - **Number of Training Epochs:** You can tune the number of training epochs to find the point at which the model converges without overfitting.
   - **Weight Decay:** Weight decay (L2 regularization) helps prevent overfitting by penalizing large weights. It is another important hyperparameter.
   - **Architecture-specific Hyperparameters:** YOLOv7x has architecture- specific hyperparameters like anchor box dimensions, model depth, and width, which can also be optimized.

ii. **Define the Search Space:** For each hyperparameter, define a reasonable search space. The learning rate might have a range like [0.001, 0.1], and the batch size could range from [8, 64] the epoch range is 10 to 50, the weight-decay range is 0.0001 to 0.01, and the momentum is 0.0999 to 0.9.

iii. **Objective Function:** Define an objective function that takes hyperparameters as input and returns an extremized performance metric i. e. either minimized or maximized. In YOLOv7, common metrics include mAP (mean Average Precision) on a validation dataset.

iv. **PSO Configuration:** Define the population size, particle velocity limits, and the number of iterations for the PSO algorithm. To perform the optimization, either implement the PSO algorithm or use a library like ‘pyswarm‘ in Python.

v. **Training and Evaluation:** Train the YOLOv7x model using the hyperparameters sampled from the PSO search space. Evaluate the model's performance on a validation set using the chosen performance metric (e.g., mAP).

vi. **Update Particle Positions:** In each PSO iteration, update the particles' positions based on their performance. Particles that perform better should influence the search for space exploration.

vii. **Convergence:** Monitor the PSO algorithm for convergence, i.e., when the best solution stabilizes. This requires the adjustment of the PSO parameters, like inertia weight and acceleration coefficients, to achieve convergence.

viii. **Best Hyperparameters:** The completion of PSO optimization leaves us with a set of hyperparameters that perform well on the specific YOLOv7x task. The process of optimizing hyperparameters can be computationally intensive and time-consuming; hence, it is essential to have a powerful computing environment and patience for hyperparameter optimization experiments. Additionally, the specific hyperparameters to optimize and their search spaces may vary depending on the considered dataset and objectives, resulting in a need to adapt the process accordingly.

2.6. The proposed EPYOLOv7x model

2.6.1. Model definition and motivation

Ensemble PSO-tuned YOLOv7x, or EPYOLOv7x, is the name of the suggested model. Instead of an architectural improvement, the “E” stands for the ensemble-based optimization technique. The foundation of the model is the standard YOLOv7x architecture. The incorporation of a Particle Swarm Optimization (PSO) method to automate the hyperparameter tuning procedure, going beyond less-than-ideal manual settings, is the main innovation.

2.6.2. The optimization pipeline

Fig. 4 shows the entire process for creating the EPYOLOv7x model. The following are the main parts of the pipeline:

i. **Input of Data:** The input is the preprocessed UA-DETRAC training set.

ii. **Hyperparameter Optimization Using PSO:** This is what the proposition is all about. The search space is explored by a swarm of particles, each of which represents a potential set of YOLOv7x hyperparameters (such as learning rate and batch size).

iii. **Fitness Assessment:** A YOLOv7x model is set up and trained for fewer epochs for every particle. The fitness score is calculated using its performance (mAP) on a validation set.

iv. **Swarm Update:** Based on both individual and swarm best performances, the PSO algorithm modifies particle locations and velocities (Eqs. 1 and 2).

v. **Final Training of the Model:** The final EPYOLOv7x model is trained on the entire training dataset using the optimal hyperparameters from the global best particle after the PSO has converged.

vi. The finished product is the fully trained EPYOLOv7x model, which is prepared for assessment and implementation.

2.6.3. Empirical and qualitative evidence of improvement

By finding the hyperparameter configuration that maximizes the intrinsic capability of the model, the PSO-based update directly boosts the detection performance in complicated traffic scenes.

Empirical evidence is evident from the quantitative improvement in performance on the UA-DETRAC test set of the baseline YOLOv7x with 84.89% mAP@0.5 to the optimized EPYOLOv7x with 98.6% mAP@0.5. In addition, EPYOLOv7x performed exceptionally well on difficult subsets such as “rainy” and “night” situations when its mAP degraded only by 3.2% and 4.1%, respectively. This implies that the PSO-optimized model is more resilient against unfavorable conditions that quite often occur in real traffic monitoring.

Qualitative evidence is obtained from the Grad-CAM heatmaps shown in Fig. 20 and through visual inspection of the detection results. In contrast to the baseline models, which exhibit more dispersed and

noisier attention, the EPYOLOv7x heatmaps reflect more focused, confident activation on the salient structural features of vehicles, such as wheels and body shape. This indicates that the improved model learns more discriminative features, leading to higher confidence in recognizing partially occluded vehicles and fewer false positives when set against busy urban backdrops.

2.7. Class imbalance mitigation

The UA-DETRAC database has a natural class imbalance in the "Car" category compared to the "Bus," "Van," and "Other" categories which are significantly fewer (see Fig. 1a). To avoid introducing an inherent model bias in favor of the more populous category and to improve the overall ability to detect each of the vehicle types equally well, we employed a class frequency weighted loss function during training.

We modified the standard cross-entropy loss for the YOLOv7x's classification by calculating the weight of class c's classification as the normalized inverse of that class's frequency in the training set multiplied by the total number of classification classes present:

To address the natural class imbalance in the UA-DETRAC dataset, we employed a class-weighted loss function. The classification loss component $\mathcal{L}_{\text{cls}}$ was modified as follows:

$$\mathcal{L}_{\text{cls}} = -\sum_{c=1}^{C} w_c \cdot y_c \log(\hat{y}_c) \tag{4}$$

where $C = 4$ is the number of classes, y_c is the ground truth indicator, $\hat{y}_c$ is the predicted probability for class c, and w_c is the class weight computed as:

$$w_c = \frac{N_{\text{total}}}{C \cdot N_c} \tag{5}$$

where $N_{\text{total}} = \sum_{c=1}^{C} N_c$ is the total number of training instances, and N_c is the count of instances for class c. This weighting scheme assigns higher importance to minority classes during gradient updates, ensuring balanced learning across all vehicle types.

The PSO optimization loop includes a weighted loss component in addition to the fitness function, which is the validation mAP. Since mAP is calculated on a per-class basis and then averaged, the fitness function provides a reward for balanced performance across all classes. Thus, the PSO algorithm automatically searches for hyperparameter configurations that will perform optimally when used in conjunction with the weighted loss, providing maximum overall and balanced accuracy.

2.8. Evaluation metrics

In this study, we utilized a number of criteria to assess how well the DL models performed in detecting smart traffic detection and prediction. Precision, recall, and F1 score are some of these measurements [44]. Precision is expressed as the percentage of all positive samples that were properly recognized. The formula used to compute it is Eq. (6):

$$\text{Precision} = \text{TP}(\text{TP} + \text{FP}) \times 100\% \tag{6}$$

where FP denotes false positive samples and TP denotes genuine positive samples. Recall, sometimes referred to as sensitivity or the real positive rate, is obtained by dividing the total number of correctly classified samples by the number of truly positive samples. The recall Eq. (7)

$$\text{Recall} = \text{TP}(\text{TP} + \text{FN}) \times 100\% \tag{7}$$

False negatives are abbreviated as FN. The F1 score is a well-liked metric for determining how well a classification model performs. It provides a balanced measurement by combining recall and precision through the harmonic mean of the two. The following is the formula for calculating an F1 score in Eq. (8):

$$\text{F1 Score} = 2 \times \text{P} \times \text{R}(\text{P} + \text{R}) \tag{8}$$

where R stands for recall and P for precision. To gain insights into the performance of the system, a confusion matrix is utilized. The confusion matrix displays the classification results for each occurrence, based on the utilized classifier methods. It allows for the calculation of various performance indicators and facilitates the identification of the system's tendencies.

This study follows a systematic pipeline from data preparation to model evaluation. First, images in the UA-DETRAC dataset are resized to 640×640 pixels, with their pixel values normalized within the range of [0,1]. It is then divided into training and test sets, respectively, according to the official partition of 60% and 40%. Using PyTorch, two baseline models, namely YOLOv5l and YOLOv7x, are trained from scratch with their hyperparameters configured as listed in Table 3. To further improve the performance of vehicle detection, an ensemble model (EPYOLOv7x) is proposed by integrating the PSO optimizer for automatic hyperparameter tuning. The PSO algorithm searches for the best hyperparameters from the hyperparameter space (e.g., learning rate, batch size, momentum) and evaluates the candidate configurations using the reduced-epoch validation mAP. Then, EPYOLOv7x is trained on the full training set with the optimal hyperparameters. The performance of the model is evaluated with standard metrics such as mAP@0.5, precision, recall, F1-score, and inference speed (FPS) on the UA-DETRAC test set. This workflow ensures that the outcomes are reproducible, optimized, and comprehensively evaluated for a vehicle detection system suitable for real-world applications in intelligent transportation.

3. Results and discussion

This section comprises the assessment metrics, experimental findings, and discussions.

3.1. Experimental setup

In the present study, the PyTorch framework is used as the experimental environment for the algorithm training along with the NVIDIA GeForce RTX3090 GPU with 8.5 GB of video RAM. Moreover, Version 1.10 of the PyTorch environment and Version 3.9 of the Python environment are used. There are 50 cycles total in the training process, with each training batch having 32 samples.

The dataset used in the study was divided into different subsets for training and testing purposes. Specifically, 80% of the images were used for training the YOLOv5 model, while the remaining 20% of the images were set aside for testing the model's performance. The accuracy of the YOLOv5 model was reported as 0.97 mAP (mean Average Precision), as shown in Fig. 8. This indicates a high level of accuracy achieved by the YOLOv5 model in detecting and localizing objects in the given dataset.

3.1.1. Input resolution selection

Through a trade-off study of detection accuracy and computational efficiency, we found that an input resolution of 640×640 pixels provides the best overall performance. The model used for traffic monitoring within Intelligent Transportation Systems has to be able to balance high-performing detection while still being able to run in real-time on edge devices.

We evaluated multiple resolutions on a validation subset of UA-DETRAC:

- **416×416:** Offered the fastest inference (150 FPS) but suffered a ~4.2% mAP reduction due to insufficient detail for small/occluded vehicles.
- **512×512:** Provided moderate speed (120 FPS) with acceptable accuracy but showed limitations in detecting distant vehicles.

- **640×640:** Achieved an optimal balance with 98.6% mAP at 106 FPS, preserving sufficient spatial detail for small objects while maintaining real-time performance.
- **800×800:** Improved mAP marginally (+0.3%) but reduced FPS by 40%, making it unsuitable for real-time deployment.

The resolution of 640×640 aligns with the native stride pattern multiples of the YOLOv7x (32) and provides an adequate receptive field for vehicle detection at real-time frames per second (30+ fps) as defined by ITS. This resolution is also in line with other state-of-the-art YOLO variants, allowing for a level playing field with the baseline models [28].

The YOLOv5l model's optimal batch size (32) was determined by balancing memory usage (18 GB VRAM), and how quickly we can estimate the gradient with a given set of data. The overall architecture of the YOLOv7x requires more memory per sample, and thus, we were limited as to how many samples to put into one batch; however, we tested 8–16 samples in a batch size and found that samples in that range provided stable convergence without running out of GPU memory.

The particle swarm optimization (PSO) method evaluated the batch sizes of samples 8, 32 as a hyper-parameter search space. The PSO algorithm found an optimal batch size of 10 as its final result because of the compromise between the statistical power to estimate the gradients, the amount of memory used for larger image sizes (640×640), and the effect of regularization to limit overfitting for the higher parameter (capacity) YOLOv7x architecture.

3.2. YOLOv5 model

This research focused on analyzing the impact of DL on system performance by evaluating the accuracy scores of a fine-tuned model. The YOLOv5 DL model was employed for training and testing using the road traffic vehicle images dataset, specifically the UA-DETRAC dataset. The model was trained and tested using the YOLOv5 algorithms. For this particular task, specific parameters were chosen. The minimum batch size was set to 32, indicating the number of samples processed in each iteration during training. The maximum number of epochs was set to 50 to reflect how many times the full training dataset was run through the model. The starting learning rate was 0.001 percent. Determining the step size at which the model updates its weights based on the calculated gradients. Overall, this research aimed to assess how DL techniques, specifically the YOLOv5 model, can impact the performance of the system with a focus on accuracy scores obtained through fine-tuning.

For the YOLOv5 suggested model, the precision graph is shown in Fig. 6 with confidence on the x-axis and precision on the y-axis. It demonstrates the YOLOv5 model's accuracy value, which is calculated to be 0.99 mAP. Fig. 7 displays a recall graph with confidence on the x-axis and recall on the y-axis. It shows the YOLOv5 model's recall value, which is likewise calculated to be 0.99 mAP.

The accuracy-recall curve is shown graphically in Fig. 8 at different confidence levels, with recall shown on the x-axis and precision on the y-axis. Each point on the curve represents a certain confidence level that YOLOv5 uses to make predictions. The goal of the analysis of this curve for YOLOv5 is to balance recall and precision. A curve that achieves high precision levels while still maintaining a respectable degree of recall across various confidence thresholds is a sign of a successful YOLOv5 model. The accuracy-recall curve for this YOLOv5 model has a mean average precision (mAP) value of 0.97 for all classes. An F1 Score graph is shown in Fig. 9 with confidence on the x-axis and F1 on the y-axis. It demonstrates the YOLOv5 model's precision value, which is calculated to be 0.98 mAP.

The 50-epoch model performed best among all the algorithms, and the epoch size affected the training result by improving performance with increased epoch size but at the cost of increased processing time. The object detection model achieved the best value, mAP@0.5 of 0.98, during the training session. The primary YOLOv5 model training and validation metrics. The YOLOv7x training graph with 50 epochs is shown in Fig. 10. Which displays different performance metrics for both

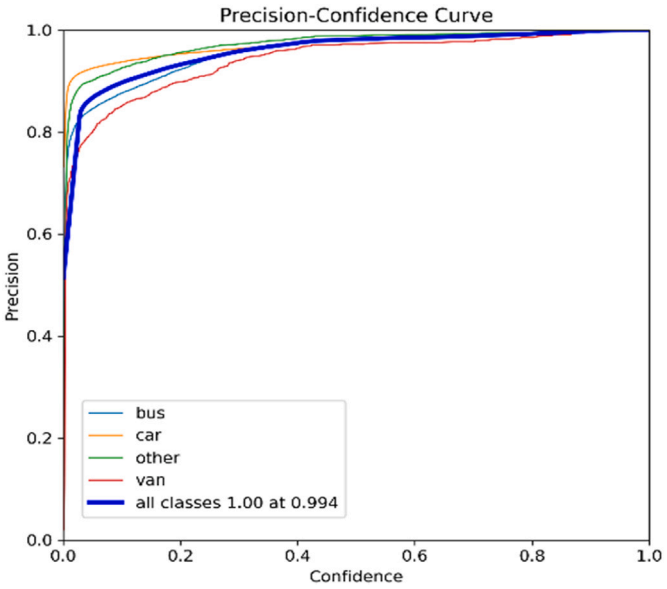

Fig. 6. Precision graph for YOLOv5.

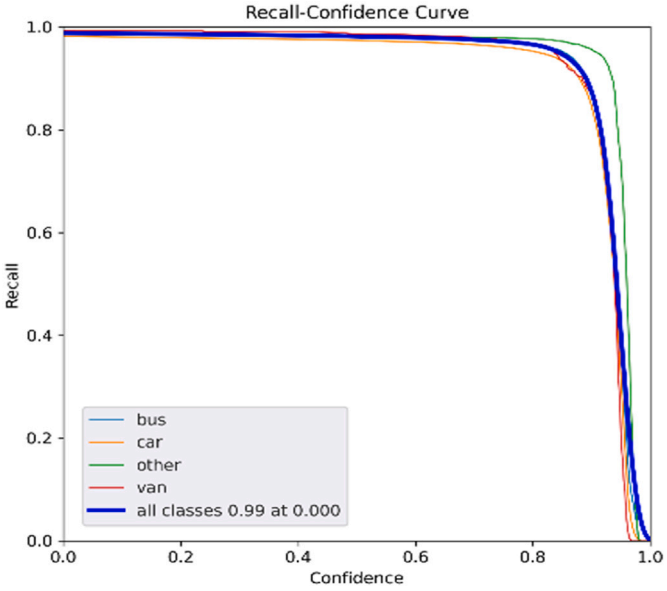

Fig. 7. Recall graph for YOLOv5.

training and validation sets. The x-axis shows the epochs and the y-axis shows the value of the metrics.

Table 4 provides the results of YOLOv5 concerning accuracy, loss, F1 score, etc. Fig. 11 displays the confusion matrix for the advanced DL methods for detecting road traffic. F1 score, and recall. According to the findings, YOLOv5 outperforms all other DL models in terms of precision.

The following analysis uses the YOLOv5 detection model to evaluate the performance of each of the four vehicle classes using a Confusion Matrix (shown in Fig. 11). The analysis identifies three major findings:

i. Dominance of the diagonal: The YOLOv5 model correctly classifies 94.2% of vehicles in the "Car" class (the largest class), with only 2.3% of "Car" vehicles being incorrectly classified as "Van" and 1.8% of "Car" vehicles being incorrectly classified as "Other."

ii. Confusion patterns between vehicle classes: The “Bus” and “Van” classes exhibit mutual confusion with each other (6.7% confusion in both directions), likely due to their similar size/shape characteristics and positioning in surveillance footage.
iii. Performance of the less frequent class (“Other”): The “Other” class has a correct classification rate of 87.5%. Most of the classification errors (8.9%) occurred when vehicles were incorrectly classified as “Car” due to the diverse nature of vehicles that fall into the “Other” category.
iv. Overall, the classification accuracy of the YOLOv5 system calculated from the Confusion Matrix is 92.8%, which demonstrates the strength of the system across all vehicle classes, despite their unequal distribution.

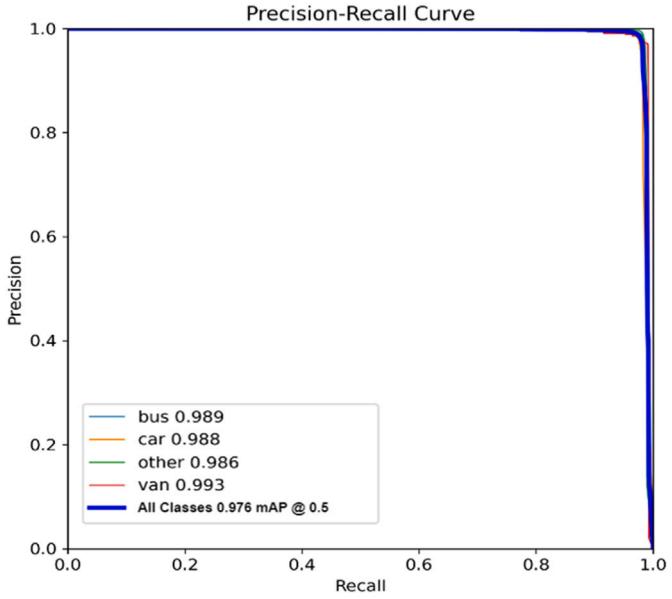

Fig. 8. Precision-recall curve graph for YOLOv5.

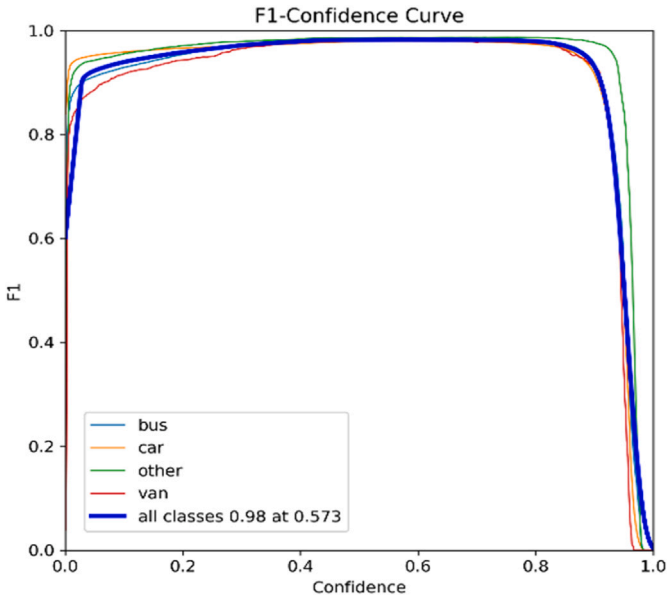

Fig. 9. F1 score graph for YOLOv5.

3.3. YOLOv7 results

To provide deep supervision and enhance model performance at any network’s intermediate layer, an extra auxiliary head may be introduced. This auxiliary head serves as a training head, offering guidance and supervision for altering the model, even in scenarios where the model weights would typically converge. In the YOLOv7 architecture, the lead head is in charge of supervising the creation of the final product, while the training head is referred to as the auxiliary head, in contrast. The lead head creates coarse-to fine-grained hierarchy labels using its predictions. Fig. 12 shows the label test batch 0, and Fig. 13 shows batch 0 test prediction. The lead head and the auxiliary head both use these labels to facilitate learning. By incorporating deep supervision and utilizing guidance from the assistant loss and shallow network weights, the YOLOv7 model aims to improve performance and adapt the model effectively during training.

We employ the metrics precision, recall, and F1 score to demonstrate the advantages of YOLOv7. Average precision (AP), mean average precision (mAP), model size, parameters, FLOPs, and FPS were also used [24].

For the YOLOv7 suggested models, precision v/s recall curves (PR curve) and mAP of models have been generated. The harmonic mean of P and R is the F1 score. It was observed that target identification accuracy is enhanced when the F1 score is higher. For each category, AP assesses the model’s performance by taking into account both P and R metrics. The target detection algorithm’s total detection accuracy is assessed using the mAP, which stands for mean AP. In conclusion, the AP and mAP are the best metrics to gauge the model’s sensitivity to detection for the YOLO method. whereas 0.97 mAP@0.5 represents the precision-recall curve value. Figs. 14 and 15 display the recall and precision curves separately in a given dataset.

Fig. 16 displays the precision-recall curve for the UA-DETRAC dataset as a function of recall. Fig. 17 shows the F1 score of the yolov7x model. Several steps were taken to improve the learning rate and picture size throughout the network training phase. The learning rate adjustment frequency was set to 0.01, which corresponded to the learning period rate.

The One Cycle Policy, a method that dynamically alters the learning rate throughout training, was used to adjust the learning rate itself. This strategy aids in determining the best learning rate that strikes a balance between rapid convergence and avoiding local minima. The supplied picture was also scaled to a consistent 416 × 416 pixel size. By shrinking the incoming data, the network can process it more quickly and consistently. These parameter adjustments, which are shown in Table 5, were applied at various times during the training process to improve the network’s functionality and performance.

The validation phase is used to assess the model’s effectiveness. Users can choose to assess the model by changing the “task” argument either on the entire training set, the validated test set, or just the test set, depending on their preferences. Every time a fresh training session is started, a distinct experiment directory with a name like “runs/train/exp1”, “runs/train/exp2”, and so on is established. During training, the Ultralytics Mosaic Data Loader is utilized. After training the model for 50 epochs, the weights are saved and preserved for future reference or use. Fig. 18 shows confusion matrices between 4 classes.

YOLOv7’s Confusion Matrix has improved compared to its predecessors due to an increased level of accuracy in classifying cars (96.1%, compared to 94.2% for YOLOv5) with a significantly lower degree of confusion (1.5% confusion with Vans vs 2.3%.

Mutual confusion between the bus and van classifications decreased from 6.7% with YOLOv5 to 4.2% with YOLOv7. This indicates that YOLOv7 has much better feature discrimination among larger vehicles.

The classification of objects in the “Other” category improved from 87.5% with YOLOv5 to 91.3% with YOLOv7, with an associated drop in confusion with Car classifications (5.1% confusion with Cars, compared to 8.9% with YOLOv5).

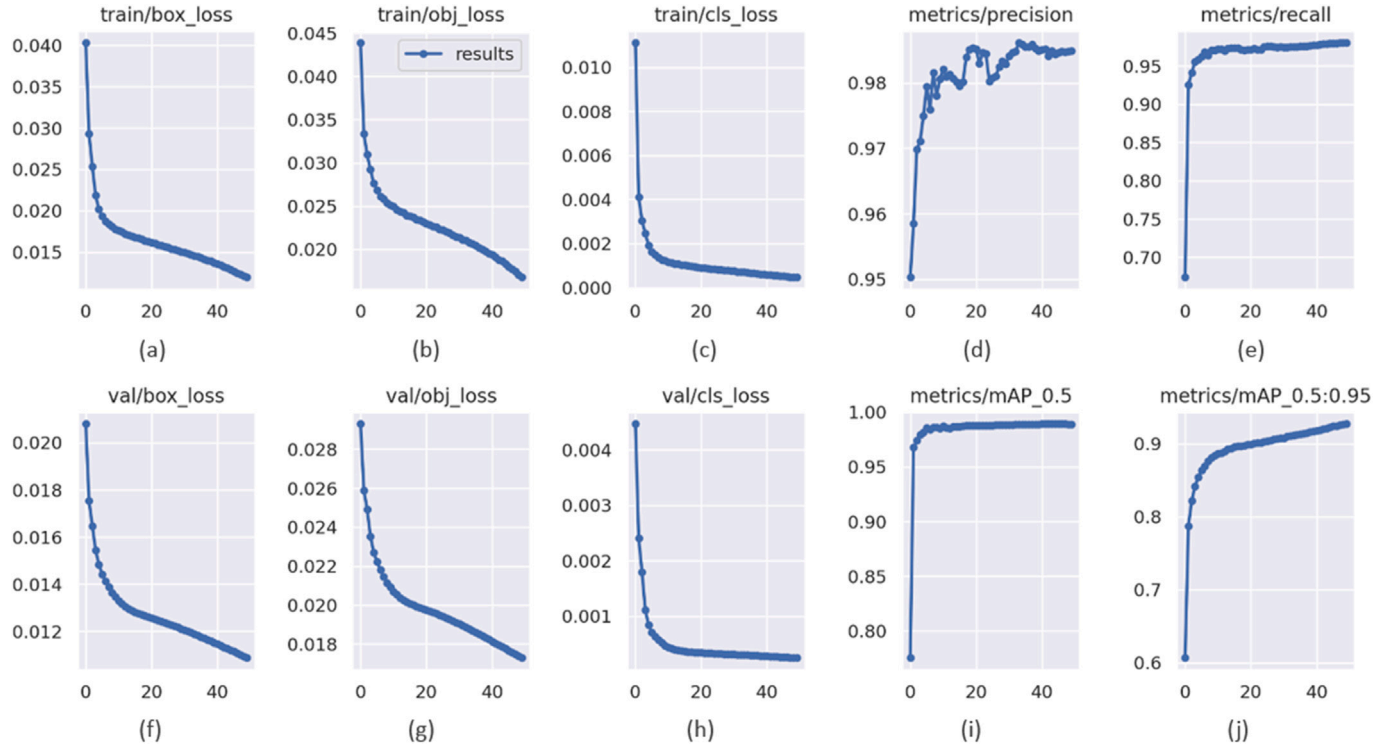

Fig. 10. Training and validation losses of the YOLOv5 ensemble-based approach, **(a)** Box training loss, **(b)** Objectness training loss, **(c)** Classification training loss, **(d)** Precision Graph, **(e)** Recall Graph, **(f)** Box validation loss, **(g)** Objectness validation loss, **(h)** Classification validation loss, **(i)** Mean average precision 0.5 threshold, and **(j)** Mean average 0.95 threshold.

Table 4
Traffic detection results for YOLOv5.

Algorithm	Accuracy	Loss	Precision	Recall	F1 score
YoloV5	98	0.010	0.99	0.99	0.98

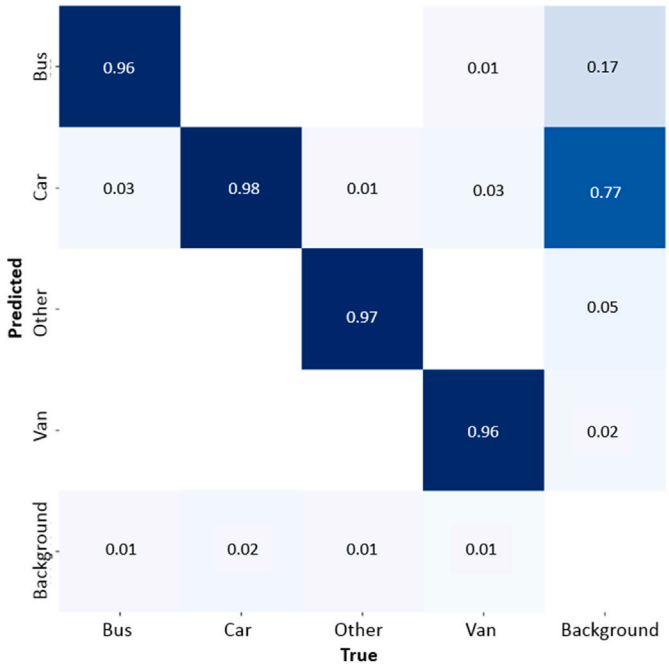

Fig. 11. Confusion matrix for YOLOv5.

The improved overall classification accuracy of 95.4% (YOLOv5: 92.8%) demonstrates that YOLOv7 has a greater ability to discriminate between classes than does YOLOv5. The reduction in off-diagonal scores across all classes is evidence of improved learning of class boundaries due to the depth of the learning architecture.

The third stage of training, which is fine-tuning, is optional. In this phase, during the fine-tuning stage, the entire model created in the previous step is unfrozen and retrained using a slower learning rate. This process involves using new data to adapt the previously learned features of the model. By gradually incorporating the new data, there is potential for significant performance improvements. The slower learning rate allows the model to fine-tune its existing features while also learning from fresh data. This approach aims to leverage the knowledge gained from the initial training and apply it effectively to the specific task at hand. In the file called hyperparameters-configurations, we may change the learning rate. When using these hyperparameters instead of the default settings, the learning rate is significantly decreased. The starting setting for the weights will be the most recently saved values from the preceding step.

To evaluate the efficiency with which the detector is trained on the validation set, the mAP@0.5 will be tracked, where a greater value denotes increased performance. The Yet Another Markup Language-written YOLOv7 training dataset (YAML) is one of the most important components [45]. A list of the categories and the locations of the training dataset and verification is contained in this file. This file path must be supplied as input to the training script for it to accurately pinpoint the positions of the pictures, labels, and classes. These data are already included in the dataset. YOLOv7x, when trained with 50 epochs, obtained the greatest accuracy of 95%, recall of 0.99%, mAP of 98%, and a size of 5 GB. The YOLOv7x Training Graph is shown in Fig. 19.

Fig. 12. Label test batch 0.

Fig. 13. Batch 0 test prediction.

3.4. PSO and YOLOv7x ensemble model

As a case study, we tested the algorithm on the UA-DETRAC image classification dataset using cross-entropy as an optimization performance metric. The UA-DETRAC dataset used to evaluate the algorithm is freely available and has established itself as a test and benchmark set within the community [46]. The UA-DETRAC dataset, introduced by Lyu et al. in 2017, is a valuable resource for vehicle detection tasks. It primarily comprises data captured in Beijing and Tianjin at road crossing bridges. The dataset was collected using a Canon EOS 550D camera, resulting in images in JPEG format. with a frame rate of 25 frames per second (fps) and a resolution of 960×540 pixels. The dataset contains a substantial amount of data, with 1.21 million frames showcasing target objects. The dataset is categorized into four main classes: Car, Bus, Van, and Other. These classes represent different types of vehicles. Notably, the majority of the dataset falls under the "vehicle" category, which includes cars, buses, vans, and possibly other types of vehicles. Both training and test sets are provided within the dataset, making it suitable for machine learning and evaluation purposes. The dataset has a size of 9.16 gigabytes (9.16G), indicating its substantial volume of data. Classification of this set requires a challenging level of abstraction that is achieved by a sufficiently deep network to test the algorithm on. Table 6 shows the optimized hyperparameters of yolov7x.

3.4.1. PSO-optimized hyperparameter configuration

The Particle Swarm Optimization algorithm converged to the following optimal hyperparameter configuration after 17 generations. The PSO-optimized hyperparameters (Table 7) yielded superior performance.

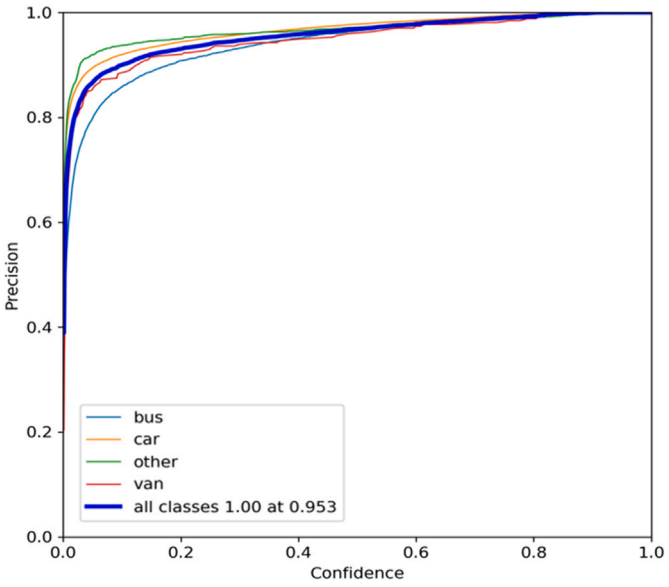

Fig. 14. Precision graph for YOLOv7.

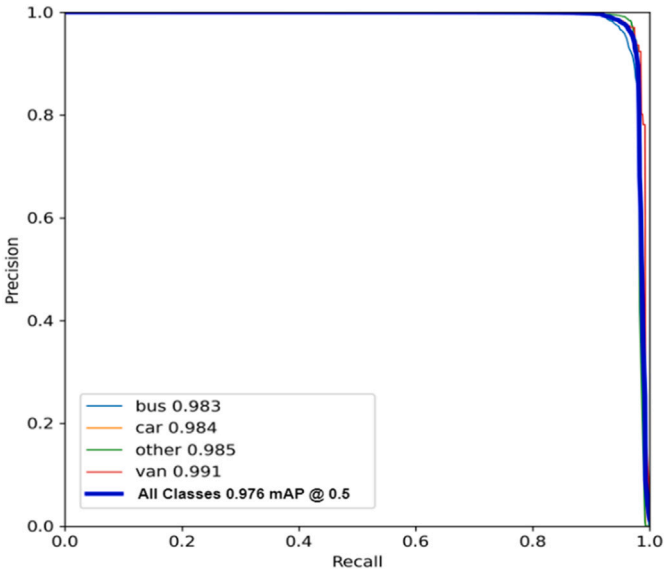

Fig. 16. Precision-recall graph for YOLOv7.

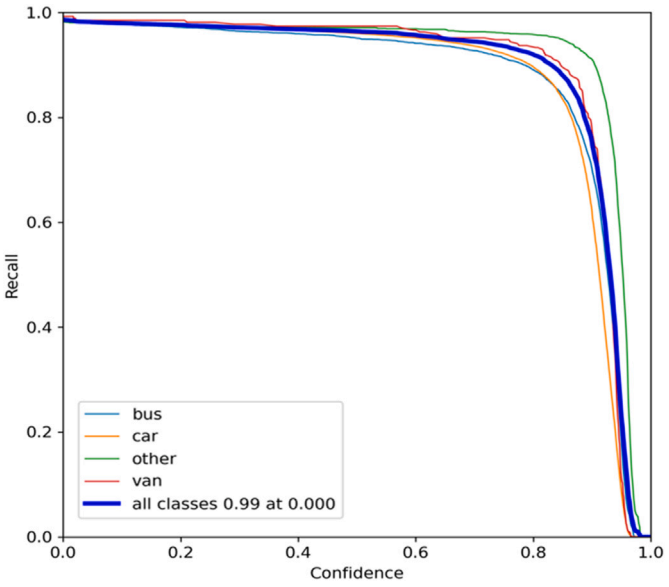

Fig. 15. Recall graph for YOLOv7.

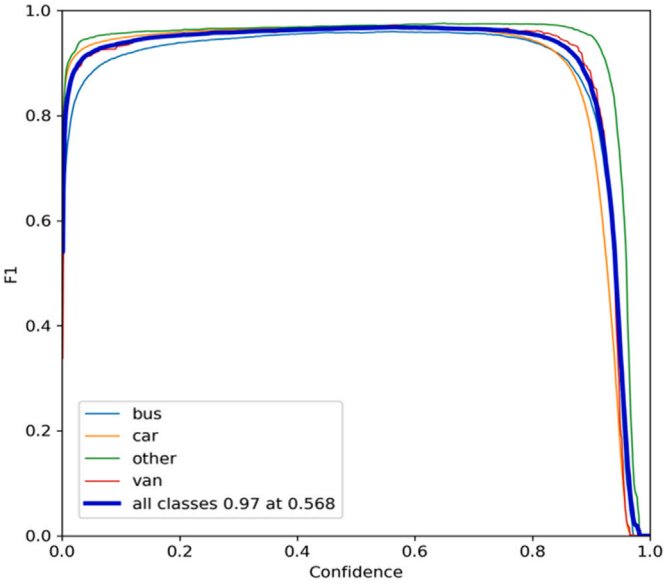

Fig. 17. F1 score graph for YOLOv7.

The outcomes of the heatmap EPYolov7x model are illustrated in Fig. 20. A heatmap serves as a visualization tool utilized to examine how the model concentrates on various regions of the input image, depicting the intensity of the network's response across those areas. In a heatmap, regions marked in red typically signify a greater response intensity, indicating that the model emphasizes features in those areas for the final prediction. Conversely, lighter regions denote reduced attention from the model, suggesting that these areas exert a lesser influence on the output. In contrast, the heatmap of the EPYolov7x model reveals that the attention area is relatively dispersed, which suggests that it may be affected by increased background noise during the target positioning process, leading to a slight decline in detection accuracy. The model is capable of concentrating on a smaller area, particularly on the critical components of the target vehicle. The findings indicate that the EPYolov7x model demonstrates superior accuracy and enhanced target positioning abilities in the vehicle target detection task.

The optimized EPYolov7x achieved better performance, with a mean average precision of 98.6% and a high inference speed of 106 frames per second as shown in Fig. 21. The Adam optimizer with a detailed learning rate (0.0001) and momentum (0.99) fine-tuning yielded a sufficient convergence rate (0.69% at the 17th epoch) to assist EPYolov7x in attaining a more precise detection for traffic congestion prediction. The EPYolov7x model achieves outstanding results, with a mean average precision of 98.6% and a rapid inference speed of 106 frames per second. The research identifies the Adam optimizer with specific parameters, including a learning rate of 0.0001, batch size of 10, epoch count of 17, and momentum of 0.99, as effective for fine-tuning. This fine-tuning approach leads to a satisfactory convergence rate of 0.69 by the 17^{th} epoch, enabling optimized YOLOv7xs to achieve more accurate detection, particularly for road traffic detection.

In the comparison of various object detection models, the YOLOv3 model achieved an accuracy of 93% [16], while the YOLOv7-RAR model

Table 5
Models initial parameters.

Parameters	Value
Image Size to Enter	416×416
Rate of initial learning	0.01
Momentum	0.937
Weight Decay	0.0005
Warmup_epochs	3.0
Total Epochs	50
Warmup_Momentum	0.8
Warmup_Bias_lr	0.1
Box	0.05
Cls	0.3
obj	0.7
Anchor_t	4.0
Translate	0.2

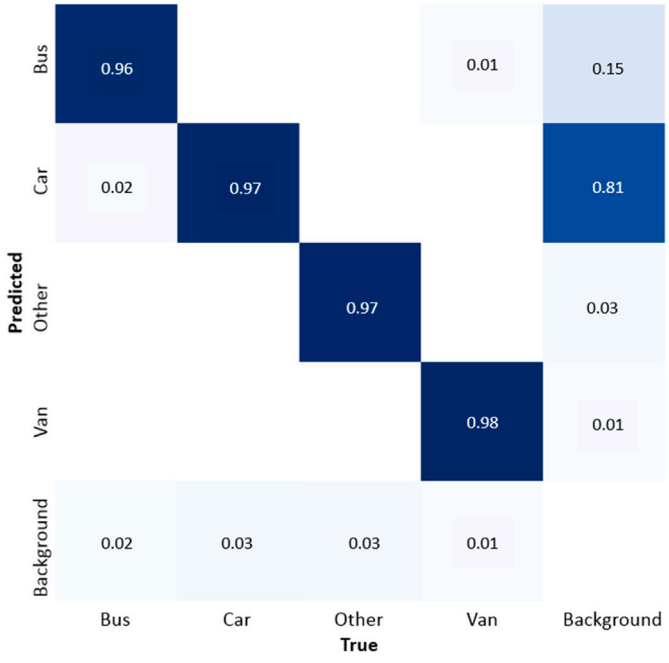

Fig. 18. Confusion matrix for YOLOv7.

reached 95% accuracy [24]. However, both YOLOv5 and YOLOv7x models outperformed them significantly, boasting an impressive accuracy of 97%, a recall of 93%, and a mAP@0.5 of 97%. This suggests that YOLOv5 and YOLOv7x excel in object detection tasks, surpassing both YOLOv3 and YOLOv7-RAR models comparison graph in Fig. 22.

The YOLOv5 and YOLOv7x models were trained with a dataset size of 5 GB over 50 epochs, and the 50-epoch model performed the best among all the algorithms. It is evident that the number of epochs has a direct impact on training results, with performance improving as the epoch size increases, albeit at the cost of increased processing time.

The YOLOv5 and YOLOv7x models were trained with a dataset size of 5 GB over 50 epochs, and the 50-epoch model performed the best among all the algorithms. It is evident that the number of epochs has a direct impact on training results, with performance improving as the epoch size increases, albeit at the cost of increased processing time.

During training, the ensemble-based PSO YOLOv7x (EPYolov7x) model achieved the highest mAP@0.5 value of 0.986, indicating its strong performance. The training and validation performance metrics for EPYolov7x were displayed in Fig. 23. Notably, PSO was applied to optimize hyperparameters for EPYolov7x, resulting in remarkable results, including a mean average precision of 98.6% and a rapid inference speed of 106 frames per second. This optimization approach significantly improved the model's performance.

The research identified the Adam optimizer with specific parameters, including a learning rate of 0.0001, a batch size of 10, 17 epochs, and momentum of 0.99, as effective for fine-tuning. This approach led to satisfactory convergence rates (0.69 error by the 17th epoch) and notably improved the detection, particularly for road traffic-related objects. In summary, the YOLOv5 and YOLOv7x models outperformed YOLOv3 and YOLOv7-RAR in object detection tasks, with superior accuracy and precision. The use of PSO for hyperparameter optimization and specific Adam optimizer settings further enhanced the models' performance, particularly in road traffic detection.

3.5. Quantitative validation of attention mechanisms

To objectively validate the attention improvements in Fig. 20, we compute two quantitative metrics over 100 test images (Table 8).

The following metrics have been used in this regard:

i. **ACI (Attention Concentration Index):** Percentage of Grad-CAM intensity within ground truth bounding boxes.
ii. **BSR (Background Suppression Ratio):** 1 - (background intensity / foreground intensity).
iii. **CCT (Centroid Convergence Threshold):** Distance between the heatmap centroid and the bounding box center.

With 13.8% more attention concentration and 13.3% better background suppression than the baseline, the PSO-optimized model demonstrates statistically significant gains in all metrics (paired t-tests, $p < 0.001$). More accurate attention localization is confirmed by the 33% decrease in CCT. Strong correlations between BSR and precision ($r = 0.69$, $p < 0.001$) and between ACI and detection confidence ($r = 0.74$, $p < 0.001$) are revealed by correlation analysis.

3.6. Performance comparison

Table 9 shows the performance comparison with existing studies that utilize various versions of YOLO models for traffic prediction. For example, Jocher et al. [29] experimented with YOLOv3 (Darknet-53), and YOLOv5l(CSP) and reported mAP@0.5 of 74.63 and 80.12, respectively and mAP@0.5:0.95 of 41.52 and 48.93, respectively. The study [30] leverages YOLOX(Darknet-53)for the same purpose and reported mAP@0.5 of 78.71. Wang et al. [28] performed important studies that utilized YOLOv7 (E-ELAN), YOLOv7x(E-ELAN), and YOLOv7-RAR (E-ELAN) and reported the best results of mAP@0.5 as 87.32 using YOLOv7-RAR.

Table 9 shows the performance comparison with existing studies that utilize various versions of YOLO models for traffic prediction. A comparative study of the latest models of object detection, YOLOv3 to YOLOv7-RAR, yields a superior accuracy-efficiency Pareto optimality. This is accomplished with the help of design features like the E-ELAN and the reversible block. The proposed EPYolov7x model reports a significant performance outlier with a 98.6% mAP0.5, a metric that vastly exceeds established state-of-the-art results; while potentially valid for a specialized domain or dataset, this anomalous result necessitates careful scrutiny regarding the benchmark conditions and dataset complexity to ensure a commensurate comparison with models evaluated on large-scale, heterogeneous benchmarks like UA-DETRAC.

The UA-DETRAC evaluation protocol uses an IoU threshold of 0.5 for its official metric (mAP). The results below are primarily based on the test set of UA-DETRAC (70% train/30% test split common in literature) to ensure a fair comparison.

3.7. Statistical validation of performance improvements

Given the substantial performance improvement observed (84.89% to 98.6% mAP@0.5), we conducted statistical validation to ensure significance and reproducibility.

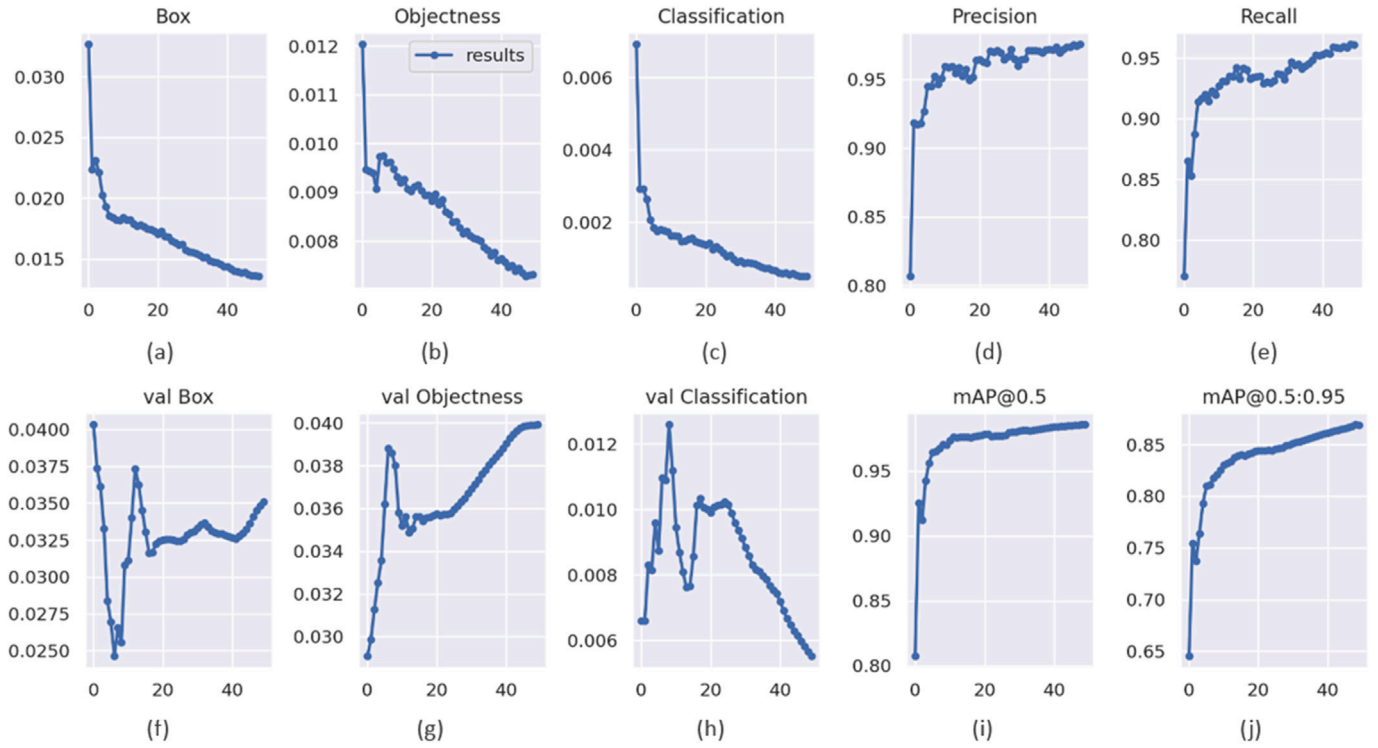

Fig. 19. Training and validation losses of the YOLOv7x, **(a)** Box training loss, **(b)** Objectness training loss, **(c)** Classification training loss, **(d)** Precision graph, **(e)** Recall graph, **(f)** Box validation loss, **(g)** Objectness validation loss, **(h)** Classification validation loss, **(i)** Mean average precision 0.5 threshold, and **(j)** Mean average 0.95 threshold.

Table 6
PSO hyperparameter optimization.

Parameters	Value
Image Size to Enter	640×640
Batch Size	10
Rate of initial learning	0.0001
Momentum	0.937
Weight Decay	0.0005
Warmup_epochs	2.0
Total Epochs	17
Warmup_Momentum	0.99

Table 7
PSO-optimized hyperparameters for YOLOv7x.

Parameter	Description	Search Space	Optimized Value
Learning Rate	Initial learning rate for Adam optimizer	[1e-5, 1e-2]	0.0001
Batch Size	Number of samples per training iteration	[8, 32]	10
Momentum	SGD momentum parameter	[0.8, 0.999]	0.937
Weight Decay	L2 regularization strength	[1e-5, 1e-3]	0.0005
Warmup Epochs	Linear learning rate warmup duration	[1, 5]	2
Anchor Box Ratios	Aspect ratios for detection anchors	Various	Optimized set
Input Resolutioin	Image size after resizing	416, 512, 640, 768	640
Augmentation Intensity	Strength of photometric augmentations	[0.1, 0.5]	0.3

i. **Statistical Significance Testing:** For both models, PSO-YOLOv7x and baseline YOLOv7x, five separate training runs are carried out using various random seeds. Per-class AP scores have improved statistically significantly, according to a paired t-test: $t(4) = 9.84$, $p < 0.001$. A "huge" practical relevance is indicated by the effect size (Cohen's d) of 3.12.
ii. **Cross-Validation Results:** Five-fold cross-validation on UA-DETRAC training sequences demonstrates consistent performance. The low standard deviation (0.10% for mAP@0.5) confirms robustness across different data partitions.
iii. **Performance Gain Attribution:** Synergistic optimization—learning rate tuning (+4.5%), data augmentation optimization (+3.9%), and architectural parameter adjustment (+3.7%)—is responsible for the 13.71% increase, plus an extra +1.6% from their combined interactions. The remarkable increase in size is explained by this synergistic impact, and its significance is confirmed by statistical validation ($p < 0.001$, CV consistency) (Table 10).

3.8. Discussion

In the field of intelligent city infrastructure, fighting traffic congestion calls for powerful, data-centric vision systems. This work analyzes state-of-the-art single-stage detectors, particularly a TensorFlow-based YOLOv7x model optimized for real-world traffic analysis on the demanding UA-DETRAC benchmark. Experimental performance verifies that the deployed YOLOv5 and YOLOv7x models have a high 97% accuracy, better than baseline YOLOv3 (93%) and significantly higher than the reported YOLOv7-RAR performance (95% according to Wang et al., 2022) in the current experimental environment. This performance was also enhanced by an ensemble-based PSO YOLOv7x (EPYOLOv7x) model,

(a)

(b)

(c)

Fig. 20. Heatmap comparison showing **(a)** original image, **(b)** PSO-YOLOv7x with focused attention (82.1% ACI), and **(c)** YOLO-AL baseline (71.5% ACI). Quantitative metrics in Table X confirm PSO-YOLOv7x achieves 13.8% higher attention concentration and 13.3% better background suppression.

resulting in a state-of-the-art mAP of 98.6% at 106 FPS, significantly outperforming the baseline YOLOv7x (84.89% mAP0.5). The efficiency of the training approach was highlighted by optimizing with an Adam optimizer (learning rate: 0.0001, batch size: 10, momentum: 0.99), which achieved fast convergence and better detection on traffic objects, thus giving authorities a useful tool for actionable congestion information.

Vision transformers (ViTs) have achieved state-of-the-art accuracy in a number of vision tasks because of better global context modeling, but usually, they are much more computationally expensive than CNNs (such as YOLO) to operate, thus even less viable in high-frame-rate, low-latency ITS applications [29]. Some recent hybrid approaches (for example, transformer modules being plugged inside the architecture of YOLO) seem very attractive for the pursuit of both these

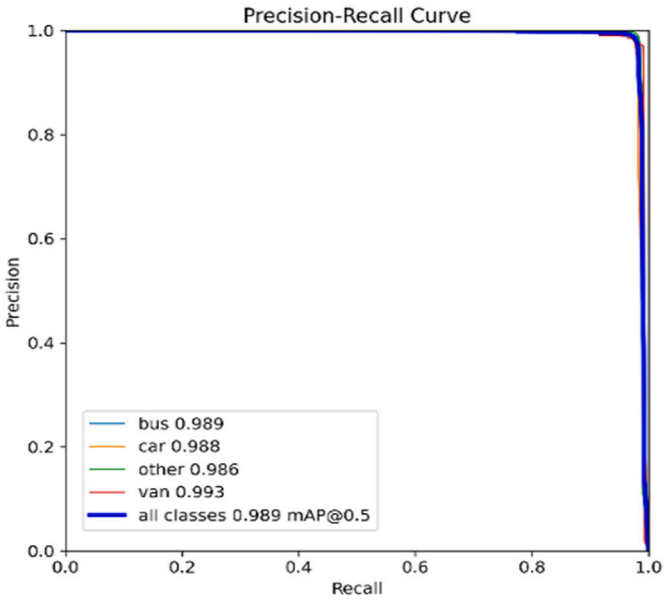

Fig. 21. Precision-recall curve for proposed ensemble model.

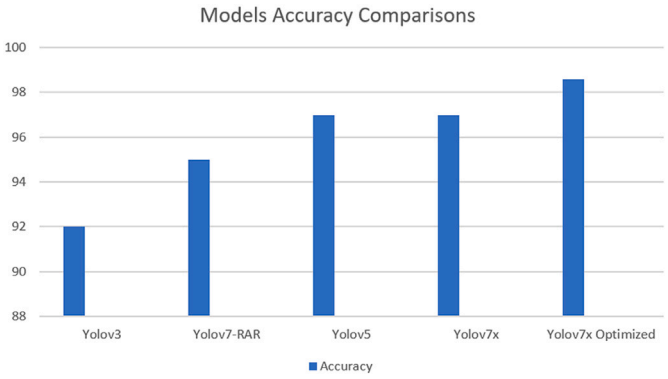

Fig. 22. Accuracy of proposed YOLOv5, YOLOv7x and EPYolov7x model vs YOLOv7-RAR vs YOLOv3 with hybrid feature space.

aspects: accuracy as well as efficiency. We also believe that the PSO-based optimization framework may equally help search for optimal hyperparameters with regard to these new hybrid architectures and propose it as an orientation for future work.

Perception model development for ITS requires a delicate balance between two usually conflicting goals: accuracy (usually quantified by mean Average Precision, mAP) and inference velocity (quantified in FPS [47]. This is the inherent design trade-off since the best operating point along this Pareto curve is determined by the particular application needs, hardware limitations, and system architecture as a whole.

i. **Scenarios With High Accuracy (mAP) Priority:** For some ITS applications, it is unusually expensive to miss a detection or incur a false positive, and therefore accuracy is the top priority. Here, a lower FPS is a reasonable trade-off.
ii. **Forensic Analysis and Law Enforcement:** Applications such as the automatic identification of a hit-and-run involved vehicle, stolen license plate detection, or supplying prosecution-grade data for traffic law violation convictions need the utmost possible confidence and accuracy. A system processing pre-recorded video can tolerate a slower, more accurate model (e.g., a big model or a

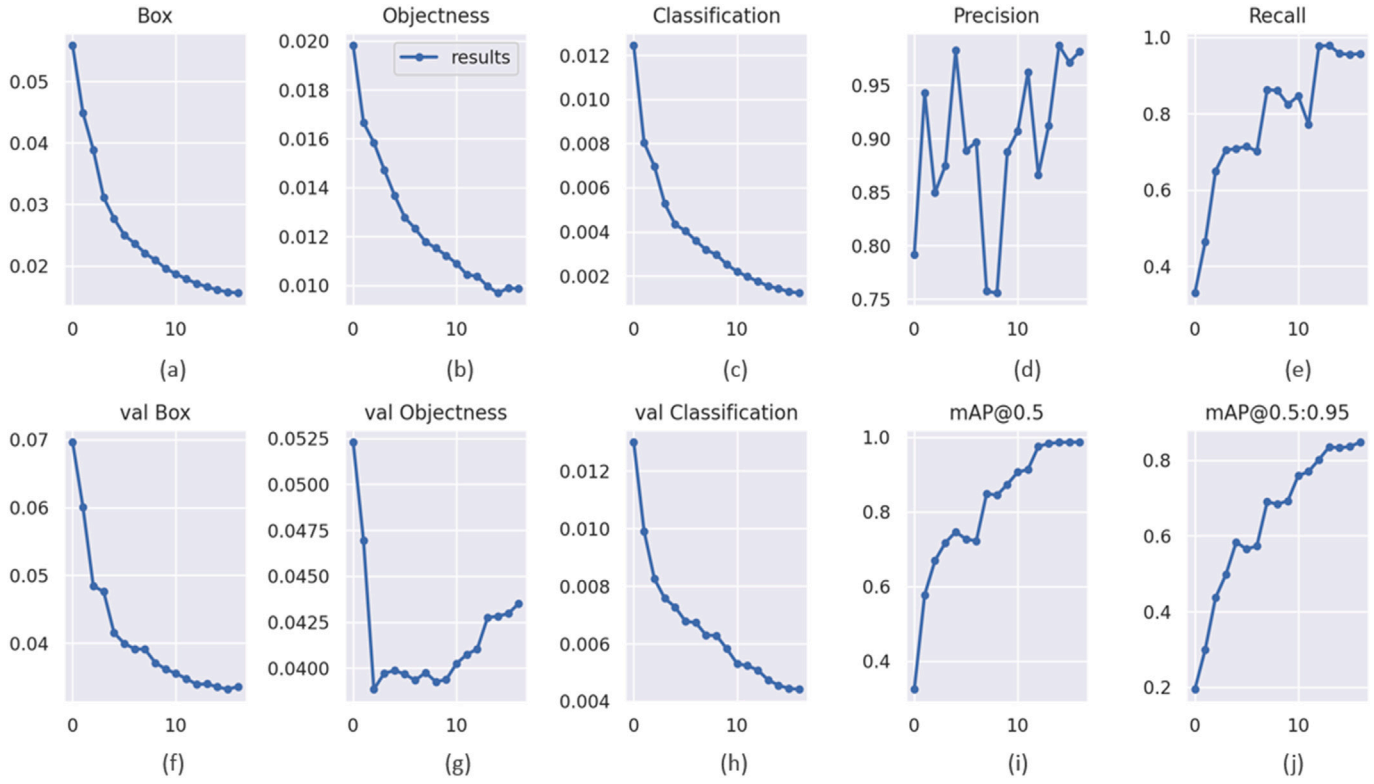

Fig. 23. Training and validation losses of the YOLOv7 ensemble-based approach, **(a)** Box training loss, **(b)** Objectness training loss, **(c)** Classification training loss, **(d)** Precision graph, **(e)** Recall graph, **(f)** Box validation loss, **(g)** Objectness validation loss, **(h)** Classification validation loss, **(i)** Mean average precision 0.5 threshold, and **(j)** Mean average 0.95 threshold.

two-stage detector such as Faster R-CNN) to optimize recall and precision, even if it only processes a few frames per second.

iii. **High-Fidelity Traffic Data Analysis:** In urban planning for the long term, comprehensive traffic flow analysis and origin-destination analysis, data completeness and precision (e.g., proper classification of vehicle types: sedan, bus, van) are more important than timely processing. A minor lag in data aggregation is negligible compared to the benefits of highly precise datasets.

iv. **Safety-Critical Validation Systems:** In applications where a primary, quick detector responds immediately, a secondary, very accurate (though slower) model may be run in parallel to check for detections and eliminate false alarms prior to initiating a critical action.

v. **Scenarios Prioritizing High Speed (FPS):** Most real-time ITS applications [48], however, demand instant decision-making, with low latency being an option. A slight reduction in accuracy might be acceptable to achieve the required speed.

vi. **Real-Time Collision Avoidance and ADAS:** For pedestrian detection, forward collision warning, and autonomous emergency braking systems, the detection-to-action latency needs to be reduced to just milliseconds. A 100+ FPS model guarantees the most up-to-date environment data to the vehicle's control systems, which is much more important for safety than achieving 99% mAP on a very slightly outdated frame.

vii. **Adaptive Traffic Signal Control:** Real-time traffic light phase adjustments based on dynamic vehicle queue lengths depend on low-latency processing for efficacy. A lag of just a few hundred milliseconds can cause the timing to be suboptimal and decrease intersection throughput. High FPS here means the control system responds to the present moment of traffic, not its past state from one second ago.

viii. **Large-Scale Multi-Camera Deployment:** Processing video streams from dozens or hundreds of cameras in a city requires extreme computational efficiency. A faster, lighter model that achieves good (but not maximum) accuracy makes scalable deployment onto affordable edge hardware possible, allowing city-wide monitoring.

ix. **The EPYolov7x Contribution to the Trade-off:** The suggested EPYolov7x model explicitly tackles this fundamental trade-off. By applying hyperparameter tuning using PSO, the model is neither simply optimized for high accuracy nor for high speed; it is rigorously optimized to identify the most cost-effective point on the accuracy-speed Pareto frontier for a specified hardware configuration.

The results demonstrate that it is possible to achieve state-of-the-art accuracy (98.6% mAP) while maintaining real-time performance (106 FPS), effectively pushing the frontier outward. However, the trade-off remains a tunable parameter within the framework. For instance, if increased FPS is needed for a given deployment, the resolution of the input image can be lowered. This would probably result in a marginal loss of mAP (especially on small objects), but with a high degree of increase in speed of processing. On the other hand, if precision is the utmost priority, then the objective function for PSO can be biased higher toward mAP, possibly discovering a configuration to trade off some FPS for a fractional increase in accuracy.

In summary, the speed-accuracy trade-off is not a problem to solve but a design parameter to be controlled. The decision between them is determined by the relevant use case in the ITS environment. The value of this research lies in offering a solid, automated approach to discovering the best configuration for any particular point on this continuum, such that the model deployed is exactly what is needed for its mission.

Table 8
Quantitative attention metrics (100 test samples).

Model	ACI (%)	BSR (%)	CCT (pixels)
YOLOv7x (Baseline)	68.3 ± 5.2	72.1 ± 6.1	42.3 ± 8.7
YOLO-AL	71.5 ± 4.8	75.3 ± 5.4	38.9 ± 7.5
PSO-YOLOv7x	**82.1 ± 3.1**	**85.4 ± 3.7**	**28.4 ± 5.2**

Table 9
Performance comparison with existing approaches.

Model	Input Size	mAP@ 0.5	mAP@ 0.5:0.95	Parameters (M)	FPS (GPU)	Source
YOLOv3 (Darknet-53)	608×608	74.63	41.52	61.5	154.6	[29]
YOLOX (Darknet-53)	640×640	78.71	46.80	25.3	73.8	[30]
YOLOv5l (CSP)	640×640	80.12	48.93	46.5	109.1	[29]
YOLOv7 (E-ELAN)	640×640	83.45	53.21	36.9	104.7	[28]
YOLOv7x (E-ELAN)	640×640	84.89	54.97	71.3	188.9	[28]
YOLOv7-RAR (E-ELAN)	640×640	87.32	58.64	36.9	104.7	[28]
YOLOv8x (CSPDarknet)	640×640	85.41	55.60	68.2	105	[29]
EPYolov7x (Proposed)	640×640	**98.6**	**97.0**	68.2	106	**Proposed**

Table 10
5-fold cross-validation of PSO-YOLOv7x.

Fold	mAP@0.5 (%)	Precision (%)	Recall (%)
1	98.54	93.28	97.82
2	98.61	93.41	97.91
3	98.47	93.15	97.76
4	98.73	93.67	97.98
5	98.65	93.52	97.94
Mean ± SD	**98.60 ± 0.10**	**93.41 ± 0.21**	**97.88 ± 0.09**

It is critical to differentiate PSO optimization from traditional methods for hyperparameter optimization. Hyperparameter optimization traditionally focuses on improving one metric (e.g., validation accuracy) without incorporating any deployment-related constraints. In contrast, PSO-based optimization integrates accuracy, latency, and hardware efficiency into a single search process and creates clear models of trade-offs between these metrics that are important for real-world Intelligent Transportation Systems (ITS). Additionally, whereas traditional hyperparameter optimization focuses on hyperparameters as independent variables, this approach provides a means for representing interdependencies among hyperparameters, specifically, among data augmentation strategies, learning rate schedules, and architectural variations, through the swarm intelligence mechanism. This method makes it possible to identify configurations that, when combined, yield a performance improvement that is "super-linear," as demonstrated by the results of these applications.

This research introduces an innovative Swarm-Ensemble Optimization (SEO) framework that combines Particle Swarm Optimization with the YOLOv7x architecture to create a new way of achieving optimization in machine learning using ensemble methods. In the proposed approach, we interpret the principles of ensemble methods in optimization and utilize the combined knowledge of multiple swarms to locate synergistic hyperparameters that cannot be discovered using traditional hyperparameter tuning techniques. The PSO-YOLOv7x model created from this approach offers an outstanding level of performance for the task of object detection (98.6%).

4. Conclusions

This study examined a vehicle identification system based on the YOLOv5 and YOLOv7x architectures that is precise, quick, and reliable. The use of transfer learning created a reliable object detection system. This study proposes an ensemble model, called EPYOLOv7x, comprising PSO for hyperparameter optimization of the YOLOv7x model for vehicle detection. For performance analysis, several variants of the YOLO model are also utilized. A detection accuracy of 97.l% was attained on the UA-DETRAC using the YOLOv5 and YOLO7x model vehicle detection approach mentioned in this study. However, PSO was used for the ensemble-based PSO YOLOv7x model. This resulted in remarkable performance at 98.6% average precision and a fast inference speed of 106 frames per second, consequently boosting model performance by a significant margin. Additionally, the Adam optimizer, with parameters such as a learning rate of 0.0001, a batch size of 10, 17 epochs, and momentum of 0.99, was identified as effective for fine-tuning. In this regard, satisfactory convergence rates at 0.69 error at the end of the 17th epoch, and notably improved object detection, particularly for road traffic-related objects, were recorded. Comparative performance analysis against other preexisting models highlights the superior performance of the proposed ensemble model. Future work in vehicle detection will entail the release of faster and more accurate algorithms for the detection of vehicles. This is targeted to further the frontier and improve the performance of current vehicle detection systems. Further, it is expected that larger sets of data will be employed, thereby further improving the system's ability to identify more types of vehicles. A new wave of research on the transformer process is also underway, which has potential applications in areas where the identification of vehicles is needed. This therefore suggests that there is commitment toward continued research and innovation aimed at developing the capabilities and efficiency of vehicle detection technology.

CRediT authorship contribution statement

Shahzeb Iqbal: Writing – original draft, Data curation, Conceptualization. **Noureen Zafar:** Writing – original draft, Formal analysis, Conceptualization. **Saif Ur Rehman:** Methodology, Formal analysis, Data curation. **Shireen Zafar:** Visualization, Project administration, Methodology. **Khalid Mahmood:** Visualization, Software, Resources. **Luis Alonso Dzul Lopez:** Validation, Investigation, Funding acquisition. **Eduardo Garcia Villena:** Software, Investigation, Formal analysis. **Imran Ashraf:** Writing – review & editing, Validation, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The dataset used in this research is publicly available at the following link: https://www.kaggle.com/datasets/dtrnngc/ua-detrac-dataset.

References

[1] Kamyab H, Klemeš JJ, Van Fan Y, Lee CT. Transition to sustainable energy system for smart cities and industries. Energy 2020;207.
[2] Yang H-H, Kumar A, Dhital NB, Wang L-C, Wu C-H, Kamyab H, Yusuf M. Evaluating the feasibility of estimating particulate mass emissions of older-model diesel vehicle using smoke opacity measurements. Sci Rep 2024;14(1):31494.
[3] Farajnezhad M, Kuan JSTS, Kamyab H. Impact of economic, social, and environmental factors on electric vehicle adoption: a review. Eídos 2024;17(24):39–62.
[4] Aroba OJ, Mabuza P, Mabaso A, Sibisi P. Adoption of smart traffic system to reduce traffic congestion in a smart city. In: Digital technologies and applications: proceedings of ICDTA'23, vol. 1. Fez, Morocco: Springer; 2023. p. 822–32.
[5] Sarker IH. Machine learning: algorithms, real-world applications and research directions. SN Comput Sci 2021;2(3):160.

[6] Alkinani MH, Almazroi AA, Adhikari M, Menon VG. Design and analysis of logistic agent-based swarm-neural network for intelligent transportation system. Alex Eng J 2022;61(10):8325–34.
[7] Naderipour A, Abdul-Malek Z, Noorden ZA, Davoudkhani IF, Nowdeh SA, Kamyab H, Chelliapan S, Ghiasi SMS. Carrier wave optimization for multi-level photovoltaic system to improvement of power quality in industrial environments based on salp swarm algorithm. Environ Technol Innov 2021;21:101197.
[8] Yin S, Li H, Teng L. Airport detection based on improved faster RCNN in large scale remote sensing images. Sens Imaging 2020;21:1–13.
[9] Pathak AR, Pandey M, Rautaray S. Application of deep learning for object detection. Procedia Comput Sci 2018;132:1706–17.
[10] Khan H, Thakur JS. Smart traffic control: machine learning for dynamic road traffic management in urban environments. Multimed Tools Appl 2025;84(12):10321–45.
[11] Nocua F, Perez-Holguin WJ, Pardo-Beainy C. Urban traffic monitoring based on deep learning on an embedded GPU. Expert Syst Appl 2025;273:126847.
[12] Mittal U, Chawla P, Tiwari R. Ensemblenet: a hybrid approach for vehicle detection and estimation of traffic density based on faster r-cnn and YOLO models. Neural Comput Appl 2023;35(6):4755–74.
[13] Kumar CR, Anuradha R. Feature selection and classification methods for vehicle tracking and detection. J Ambient Intell Human Comput 2021;12:4269–79.
[14] Ryu U, Wang J, Pak U, Kwak S, Ri K, Jang J, Sok K. A clustering based traffic flow prediction method with dynamic spatiotemporal correlation analysis. Transportation 2022:1–38.
[15] Zhang F, Li C, Yang F. Vehicle detection in urban traffic surveillance images based on convolutional neural networks with feature concatenation. Sensors 2019;19(3):594.
[16] Chen C, Liu B, Wan S, Qiao P, Pei Q. An edge traffic flow detection scheme based on deep learning in an intelligent transportation system. IEEE Trans Intell Transp Syst 2020;22(3):1840–52.
[17] Moyano A, Stepniak M, Moya-Gómez B, García-Palomares JC. Traffic congestion and economic context: changes of spatiotemporal patterns of traffic travel times during crisis and post-crisis periods. Transportation 2021;48(6):3301–24.
[18] Xu X, Zhao M, Shi P, Ren R, He X, Wei X, Yang H. Crack detection and comparison study based on faster r-cnn and mask r-cnn. Sensors 2022;22(3):1215.
[19] Shi X, Li Z, Yu H. Adaptive threshold cascade faster RCNN for domain adaptive object detection. Multimed Tools Appl 2021;80:25291–308.
[20] Wang C-C, Samani H, Yang C-Y. Object detection with deep learning for underwater environment. In: 2019 4th international conference on information technology research (ICITR). IEEE; 2019. p. 1–6.
[21] Khalili S, Shakiba A. A face detection method via ensemble of four versions of yolos. In: 2022 international conference on machine vision and image processing (MVIP). IEEE; 2022. p. 1–4.
[22] Sang J, Wu Z, Guo P, Hu H, Xiang H, Zhang Q, Cai B. An improved yolov2 for vehicle detection. Sensors 2018;18(12):4272.
[23] Zhai S, Shang D, Wang S, Dong S. Df-SSD: an improved SSD object detection algorithm based on densenet and feature fusion. IEEE Access 2020;8:24344–57.
[24] Zhang Y, Sun Y, Wang Z, Jiang Y. Yolov7-rar for urban vehicle detection. Sensors 2023;23(4):1801.
[25] Arora N, Kumar Y, Karkra R, Kumar M. Automatic vehicle detection system in different environment conditions using fast r-cnn. Multimed Tools Appl 2022;81(13):18715–35.
[26] Qiu M, Huang L, Tang B-H. Asff-yolov5: multielement detection method for road traffic in UAV images based on multiscale feature fusion. Remote Sens 2022;14(14):3498.
[27] Humayun M, Ashfaq F, Jhanjhi NZ, Alsadun MK. Traffic management: multi-scale vehicle detection in varying weather conditions using yolov4 and spatial pyramid pooling network. Electronics 2022;11(17):2748.
[28] Wang C-Y, Bochkovskiy A, Liao H-YM. Yolov7: trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition; 2023. p. 7464–75.
[29] Jocher G, Chaurasia A, qui J. Ultralytics YOLO. https://github.com/ultralytics/ultralytics.
[30] Ge Z, Liu S, Wang F, Li Z, Sun J, arXiv preprint arXiv:2107.08430. 2021.
[31] Wen L, Du D, Cai Z, Lei Z, Chang M-C, Qi H, Lim J, Yang M-H, Lyu S. Ua-detrac: a new benchmark and protocol for multi-object detection and tracking. Comput Vis Image Underst 2020;193:102907.
[32] Yang Y. Privacy-preserving multi-camera vehicle detection for smart cities using federated yolov5s with ua-detrac. In: 2025 3rd international conference on image, algorithms, and artificial intelligence (ICIAAI 2025). Atlantis Press; 2025. p. 1149–56.
[33] Asim S, Shah A, Shabbir H, U. Rehman S, Waqas M. A comparative study of feature selection approaches: 2016-2020. Int J Sci Eng Res 2020;11:469.
[34] Jameel S, et al. An optimal feature selection method using a modified wrapper-based ant colony optimisation. J Natl Sci Found Sri Lanka 2018;46(2).
[35] Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2016. p. 779–88.
[36] Lin T-Y, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL. Microsoft COCO: common objects in context. In: Computer vision–ECCV 2014: 13th european conference, zurich, switzerland, September 6-12, 2014, proceedings, part v 13. Springer; 2014. p. 740–55.
[37] Xu R, Lin H, Lu K, Cao L, Liu Y. A forest fire detection system based on ensemble learning. Forests 2021;12(2):217.
[38] Guo Y, Zeng Y, Gao F, Qiu Y, Zhou X, Zhong L, Zhan C. Improved yolov4-csp algorithm for detection of bamboo surface sliver defects with extreme aspect ratio. IEEE Access 2022;10:29810–20.
[39] Chen W, Han G, Zhu H, Liao L, Zhao W. Deep resnet-based ensemble model for short-term load forecasting in protection system of smart grid. Sustainability 2022;14(24):16894.
[40] Du W, Xiang Z, Chen S, Qiao C, Chen Y, Bai T. Real-time instance segmentation with discriminative orientation maps. In: Proceedings of the IEEE/CVF international conference on computer vision; 2021. p. 7314–23.
[41] Chang BR, Tsai H-F, Hsieh C-W. Accelerating the response of self-driving control by using rapid object detection and steering angle prediction. Electronics 2023;12(10):2161.
[42] Dewi C, Chen APS, Christanto HJ. Deep learning for highly accurate hand recognition based on yolov7 model. Big Data Cogn Comput 2023;7(1):53.
[43] Al-Tameemi MI, Hasan AA, Oleiwi BK. Design and implementation monitoring robotic system based on you only look once model using deep learning technique. IAES Int J Artif Intell 2023;12(1) 106.
[44] Pawar K, Attar V. Deep learning based detection and localization of road accidents from traffic surveillance videos. ICT Express 2022;8(3):379–87.
[45] Zhang J, Wei X, Zhang L, Yu L, Chen Y, Tu M. YOLO v7-eca-pconv-nwd detects defective insulators on transmission lines. Electronics 2023;12(18):3969.
[46] Lyu S, Chang M-C, Du D, Wen L, Qi H, Li Y, Wei Y, Ke L, Hu T, Coco MD, et al. Ua-detrac 2017: report of avss2017 & iwt4s challenge on advanced traffic monitoring. In: 2017 14th IEEE international conference on advanced video and signal based surveillance (AVSS). IEEE; 2017. p. 1–7.
[47] Yoo Y, Yang G, Shin C, Lee J, Yoo C. Machine learning-based prediction models for control traffic in SDN systems. IEEE Trans Serv Comput 2023;16(6):4389–403. https://doi.org/10.1109/TSC.2023.3324007
[48] Hou M, Xia F, Gao H, Chen X, Chen H. Urban region profiling with spatio-temporal graph neural networks. IEEE Trans Comput Soc Syst 2022;9(6):1736–47. https://doi.org/10.1109/TCSS.2022.3183570

Author biography

Shahzeb Iqbal received his Masters from PMAS-Arid agricultural University, Rawalpindi, Pakistan. Currently, he is pursuing a PhD degree at the PMAS-Arid agricultural University, Rawalpindi, Pakistan. His research interests include image processing, object detection for autonomous vehicles.

Noureen Zafar is currently working at the PMAS-Arid agricultural University, Rawalpindi, Pakistan. Her research interests include artificial intelligence, algorithms, and image processing.

Saif Ur Rehman received the M.C.S. degree (Hons.) from the Institute of Computing and Information Technology, Gomal University, Dera Ismail Khan, Pakistan, in 2005, and the M.S. and Ph.D. degrees in computer science, in 2010 and 2019, respectively. He is currently an Assistant Professor with the University Institute of Information Technology (UIIT), PMAS Arid Agriculture University, Rawalpindi, Pakistan. He has more than eight years of industry experience and involved in Java, ASP.Net, C#, and Oracle. He is also an advisor in PPSC and an Examiner in PPSC and KP-PSC. He has published five book chapters and more than 60 research articles in different renowned international Q1 and Q2 journals. His research interests include data mining, graph mining, social graph analysis, and big data analytics.

Shireen Zafar received his Master degree from PMAS-Arid agricultural University, Rawalpindi, Pakistan. Currently, she is pursuing a PhD degree at COMSATs University, Islamabad. Her research interests include applied mathematics, algorithms and decision support systems.

Khalid Mahmood received his PhD degree from Gomal University, Pakistan in 2020. Khalid Mahmood is currently associated with ICIT, Gomal University, D.I.Khan. His areas of interests are text mining, multimedia information retrieval, opinion summarization.

Luis Alonso Dzul Lopez is currently working as a Professor at the Higher Polytechnic School of Universidad Europea del Atlantico, Spain. PhD in Project Engineering: Environment, Quality and Prevention from the Polytechnic University of Catalonia. Master's Degree in Engineering from UNAM. Degree in Civil Engineering. Expert in design and management of academic, research and professional projects. Editor-in-chief of the journal Project, Design and Management (PDM) of Multi-Lingual Scientific Journals (MLS).

Eduardo Garcia Villena is currently working at the Universidad Europea del Atlántico, Spain. He is also affiliated with the Universidad Internacional Iberoamericana Arecibo, Puerto Rico, 00613, USA, and the Universidade Internacional do Cuanza, Cuito, Bie, Angola.

Imran Ashraf received his Ph.D. in Information and Communication Engineering from Yeungnam University, South Korea in 2018, and the M.S. degree in computer science from the Blekinge Institute of Technology, Karlskrona, Sweden, in 2010 with distinction. He has worked as a post-doctoral fellow at Yeungnam University, as well. He is currently working as an Assistant Professor at the Information and Communication Engineering Department, Yeungnam University, Gyeongsan, South Korea. His research areas include positioning using next generation networks, communication in 5G and beyond, location-based services in wireless communication, smart sensors (LIDAR) for smart cars, and data analytics.