Research Square

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

A Deep Learning Based Aggregative Framework for Object Detection in Road Environments

thayyaba khatoon mohammed

thayyaba.khatoon16@gmail.com

VNR vignana Jyothi institute of engineering and technology

Research Article

Keywords: Autonomous driving initiatives, Computer vision, deep learning, Object detection, Fast R-CNN, YOLO

Posted Date: March 6th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-8793529/v1

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

Additional Declarations: No competing interests reported.

A Deep Learning Based Aggregative Framework for Object Detection in Road Environments

Thayyaba Khatoon Mohammed[1]

[1]Department of Computer Science and Engineering, VNR Vignana Jyothi Institute of Engineering &Technology, Hyderabad, India.
Corresponding authors. *Email ID*: Thayyaba.khatoon16@gmail.com

ABSTRACT: In recent years, object detection has become a focal point in research due to its crucial role in video analysis and image understanding. Traditional approaches to object detection often relied on manually crafted features and shallow architectures, which led to limitations in performance and periods of stagnation. These methods typically used complex ensembles that combined various low-level image features with higher-level context from object detectors and scene classifiers. However, the rapid advancements in deep learning have introduced more powerful tools capable of learning semantic, high-level features that address the limitations of traditional architectures. This work presents a novel approach that integrates two prominent object detection methodologies with an optimization technique: Fast Region-based Convolutional Neural Networks (Fast R-CNN) and the You Only Look Once (YOLOv8) model, enhanced with Practical Swarm Optimization (PSO). Specifically, we merge the Feature Pyramid Network (FPN) and the Region Proposal Network (RPN) from Fast R-CNN to improve feature extraction and region proposal generation. Furthermore, YOLO represents a significant leap in object detection, offering a powerful blend of speed and accuracy. The proposed method performs qualitative and quantitative evaluation on standard datasets such as KITTI and in-house dataset. As a result, the proposed approach demonstrates a significant increase in quantitative metrics for object detection with Average Precision (AP) of 98.33%, and Average Heading Similarity (AHS) of 97.33%. Our results demonstrate significant improvements in detection accuracy, especially in complex scenarios encountered in autonomous driving. Moreover, our framework achieves remarkable efficiency, making it suitable for real-time deployment in practical settings.

Keywords: Autonomous driving initiatives, Computer vision, deep learning, Object detection, Fast R-CNN, YOLO.

1. INTRODUCTION

In recent times, there has been a notable transition within the automotive sector towards autonomous driving technology, spurred by remarkable progress in artificial intelligence and deep learning [1]. Within this landscape, object detection emerges as a critical component for autonomous vehicles, enabling precise identification and localization of key elements such as pedestrians and obstacles. This capability forms the foundation of ensuring the efficacy and safety of autonomous vehicles on the road [2]. Despite significant advancements in roadway safety and vehicle design, the incidence of fatal crashes continues to rise. According to the latest report from the World Health Organization (WHO) [3], road traffic accidents claim approximately 1.35 million lives annually worldwide. In response to these challenges, extensive research efforts over the past two decades have been dedicated to understanding the impact of roadway features on safety. In the realm of object detection, the problem is defined as determining the location of objects within an image [4] and categorizing each object [5].

Traditionally, object detection models follow a pipeline consisting of three main stages: selection of region information, feature extraction, and classification. This structured approach aims to identify regions of interest, extract relevant features from those regions, and then classify the objects [6]. In the realm of object detection, the process of selecting informative regions is crucial for accurately identifying objects within images. Objects can appear at various positions, sizes, and aspect ratios, necessitating a comprehensive approach to scanning the entire image [7]. This is often achieved through a multi-scale sliding window technique. However, while this approach can uncover all potential object positions, its drawbacks are evident. The computational cost is substantial due to the large number of candidate windows generated, and redundancy may arise, diminishing efficiency. Conversely, utilizing a fixed number of sliding window templates might overlook important regions, potentially leading to unsatisfactory detection results. Effective feature extraction is fundamental to the recognition of diverse objects. Techniques such as SIFT, HOG, and Haar-like features are favoured for their ability to provide semantic and robust representations [8]. These features emulate the intricate workings of complex cells in the human brain, enabling a deeper understanding of object characteristics. However, designing feature descriptors that can adequately capture the diverse appearances, varied illumination conditions, and complex backgrounds of objects presents a significant

challenge [9]. Furthermore, classification plays a pivotal role in distinguishing target objects from their surroundings. Commonly employed classifiers such as SVM, AdaBoost, and Deformable Part-based Model (DPM) play a pivotal role in advancing the capabilities of visual recognition systems. However, among these classifiers, the DPM distinguishes itself for its exceptional flexibility and robustness in handling complex object deformations. DPM operates by effectively combining various object parts with associated deformation costs, allowing it to adapt to a wide range of object shapes and configurations [10]. This adaptive approach significantly enhances the accuracy and reliability of object detection, particularly in scenarios where objects undergo substantial variations in appearance and structure.

Moreover, DPM leverages the power of graphical models, which enable the representation of complex relationships and dependencies between different object parts. By incorporating carefully engineered low-level features and kinematically inspired part decompositions, DPM achieves a holistic understanding of object structures, leading to more precise and comprehensive detection outcomes [11]. Furthermore, the utilization of discriminative learning techniques empowers DPM to construct high-precision part-based models tailored to diverse object classes. Through extensive training on annotated datasets, DPM learns to discriminate between different object categories with remarkable accuracy, thereby enhancing the effectiveness of the classification process [12]. In essence, the combination of flexible deformation modelling, graphical representations, and discriminative learning techniques endows the DPM with robust detection capabilities across a wide spectrum of object types and environmental conditions. This comprehensive approach significantly contributes to the advancement of object detection systems, enabling more accurate and reliable visual recognition in various real-world applications. However, ensuring a proper connection and logical flow between these stages Figure 1 is essential for the overall effectiveness of the object detection process. Table 1 represents the list of abbreviations used in the article.

Table 1. List of abbreviations

Term	Abbreviations
Aggregative View of Object Detection	AVOD
Light Detecting and Ranging	LIDAR
Region Proposal Network	RPN
Scale-Invariant Feature Transform	SIFT
First Information Report	FIR
World Health Organization	WHO
Scale-Invariant Feature Transform	SIFT
Histogram of Oriented Gradient	HOG
Support Vector Machine	SVM
Deformable Parts Models	DPM
Convolution Neural Network	CNN
You Only Look Once	YOLO
General Motors	GM
Deep Neural Network	DNN
Feature Proposal Network	FPN
Radio Detection and Ranging	RADAR
Region Of Interest	ROI
Fully Connected	FC
Particle Swarm Optimization	PSO
Prior personal best Position	P best
Global best position	G best

Traditional methods for collecting and analysing roadway features data have long been recognized as resource-intensive and time-consuming. Recognizing this challenge, this paper aims to leverage the potential of AI computer vision, specifically deep learning techniques, to automate the detection of roadway features. By doing so, it seeks to enhance roadway safety management efficiently and cost-effectively [13]. The proposed model introduces Fast R-CNN, which marks a significant advancement in roadway feature detection. Fast R-CNN effectively integrates CNNs to analyse specific regions of interest within images, allowing for precise detection of roadway features. This integration of deep learning techniques enables the model to learn complex patterns and variations in roadway features, leading to improved accuracy and reliability in detection tasks. Since the inception of Fast R-CNN, several enhanced models have emerged in the field of object detection. These include Fast R-CNN, which streamlines the detection process by jointly optimizing classification and bounding box regression tasks. Faster R-CNN introduces an additional sub-network to generate region proposals more efficiently, further enhancing detection speed and accuracy. YOLO represents another notable advancement, achieving object detection through a fixed-grid regression approach, thereby enabling real-time detection with high accuracy.

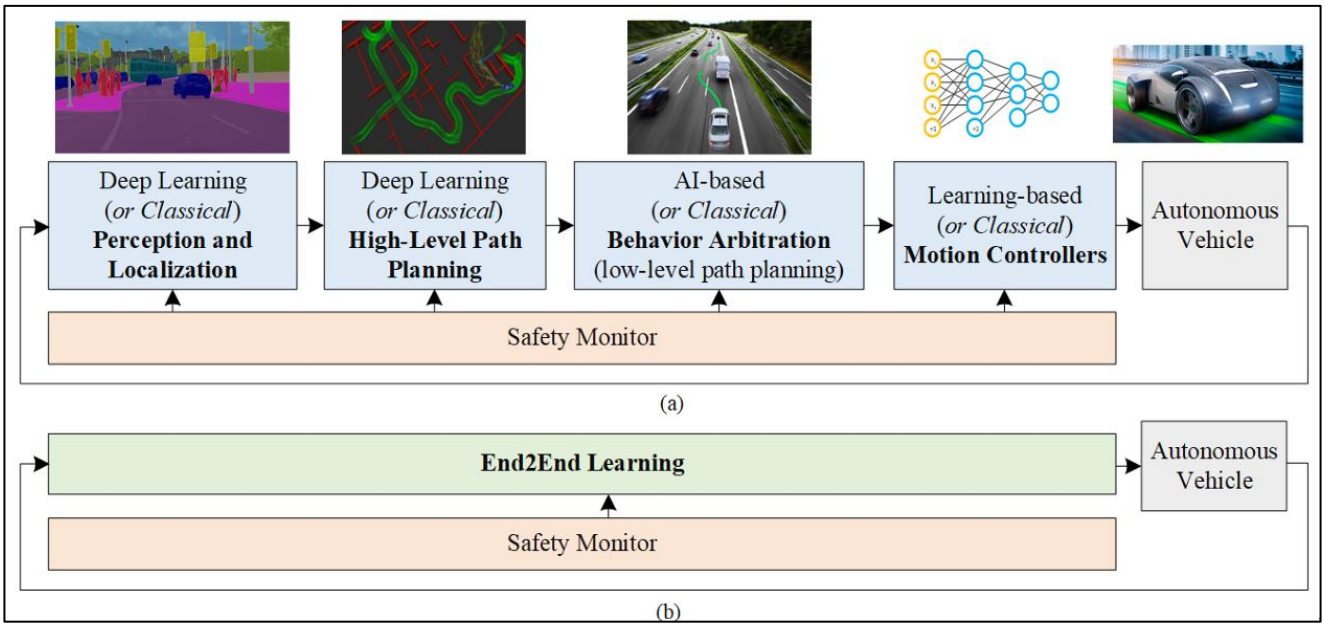


Figure.1 Stands for an autonomous vehicle that relies on deep learning. (a) an end-to-end system, (b) in the scenario of a sequential pipeline, the parts can be created utilizing artificial intelligence and deep learning techniques, or they can be based on traditional non-learning methods.

Collectively, these improved models build upon the foundation laid by R-CNN, each offering distinct enhancements to detection performance. By incorporating advanced deep learning techniques, such as joint optimization, improved region proposal generation, and fixed-grid regression, these models make real-time and accurate object detection more achievable than ever before. This progress holds immense promise for enhancing roadway safety management by automating the detection of critical features, ultimately contributing to safer and more efficient transportation systems.

The rest of this paper is organized as follows. Section 2 discuss the related work. The methodology and the flow of work is explained in Section 3. Section 4 consist of results and discussion and Section 5 concludes the paper with future directions.

2. Related Work

The concept of self-driving cars traces back nearly 80 years, originating from GM presentation at the 1939 World's Fair in New York [14]. Since then, contemporary advancements in communication networks, wireless connectivity, and sensor technology have propelled the evolution of autonomous driving systems [15]. Various components are integrated into these systems to facilitate the transportation of materials, payloads, commodities, and passengers. These components include software-defined signal processing, 3D scene analysis, network-enabled controls, human-machine interface applications, and sensor data fusion [16, 17, 18]. The capacity to operate faultlessly in a variety of settings is crucial to the success of autonomous vehicles powered by artificial intelligence [19]. To do this, it must be able to pinpoint its exact location, gather ordinary data, create fused datasets, and communicate without interruption with other cars and smart infrastructure [20]. To put it simply, these technological developments are coming together to make autonomous vehicles a reality, ones that can drive themselves safely and efficiently in different environments. This is a major step forward in the development of transportation systems, made possible by deep learning. Feature extraction, representation, and transformation are all performed by deep learning algorithms, a subset of machine learning algorithms in artificial intelligence. These algorithms use multiple layers of nonlinear processing units (neurons) [21].

Researchers and engineers have shown a great deal of interest in CNNs [22, 23], a subset of Deep Neural Networks (DNNs), owing to the proliferation of large-scale real-world data and improved computing power in the past decade. These days, it's the go-to choose for developing cutting-edge algorithms and methods for use in computer vision systems. Complex tasks like object detection and image categorization are accomplished by these systems through the use of numerous layers of neurons. Then, these algorithms can be supervised to detect objects or classify images, or trained unsupervised to analyse patterns [24]. As autonomous driving technology races towards the future, accurate object detection remains paramount for ensuring safe and reliable navigation. Traditional methods often succumb to occlusions, varying lighting, and complex backgrounds, posing safety concerns. Deep learning techniques, however, offer a powerful solution, as demonstrated by the AVOD framework [25]. AVOD's strength lies in aggregating predictions from multiple CNNs. Trained on diverse resolutions and receptive fields, these networks capture rich image information. The subsequent fusion through an aggregation module yields robust and highly accurate detection performance.

Table 2. Represents some proposed algorithms to solve object detection problems.

Reference	Year	Methodology	Dataset	Limitations
Long et al. [23]	2019	Faster R-CNN & IRP	Cityscapes	Requires a significant computational resource, particularly for real-time applications
Liu et al. [24]	2019	YOLO	Pascal VOC	Struggle to detect small objects accurately due to its single pass architecture. Limited in handling highly occluded or overlapping objects
Wu et al. [25]	2020	SSD	COCO	Struggles with small object detection and might have reduced performance in heavily cluttered scenes.
Li et al. [26]	2021	Feature Pyramid Networks (FPN)	COCO	Computational overhead due to multi-scale feature fusion, impacting real-time performance
Ding et al. [27]	2021	Faster R-CNN	ImageNet	Increased complexity compared to Faster R-CNN, leading to higher computational costs
Kumar et al. [28]	2022	SOLO	Cityscapes	Uses extensive computational power for pixel-level segmentation. Challenges with precise object boundaries and heavily obscured instances.
Gong et al. [29]	2022	R-CNN	KITTI	Increased computational complexity due to cascaded refinement stages, impacting real-time performance. May require careful tuning of cascade thresholds.
Shen et al. [30]	2023	Pyramid Anchor Attention (PAA)	Open Images Dataset	Potential computational overhead due to multi-scale attention mechanism. Effectiveness may vary depending on the dataset and object characteristics.
Qin et al. [31]	2023	Meta-learning framework	KITTI	Relies on meta-training data to achieve adaptability, may require substantial computational resources for meta-learning phase
Deng et al. [32]	2023	Attentive region proposal network	Open Images Dataset	Additional computational overhead due to attention mechanism, effectiveness may vary depending on dataset and object characteristics
Yang et al. [33]	2024	Augmented object detection	Pascal VOC	Requires additional computational resources for training and deploying generative models for data augmentation, potential challenges in ensuring the quality and diversity of augmented data
Pang et al. [34]	2024	YOLO	KITTI	May struggle with small object detection and precise localization due to its single-pass architecture.

Deep learning further enhances AVOD's capabilities. FPNs [35] enable effective handling of objects of varying sizes. Attention mechanisms [36] focus on crucial image regions, improving accuracy in challenging scenarios. Data augmentation techniques like random cropping and colour jittering artificially diversify the training data, boosting the model's ability to generalize to unseen situations [37]. The results are promising. AVOD, integrated with these deep learning techniques, has achieved state-of-the-art results on popular benchmarks like KITTI, surpassing traditional and other deep learning-based detectors [38]. Additionally, studies highlight its efficiency gains compared to ensemble-based methods [39]. However, challenges remain. Real-time performance, robustness to sensor noise, and explainability of detection decisions require further exploration. Integrating AVOD with other autonomous driving functionalities, such as tracking and path planning, is an ongoing effort [40]. Despite these challenges, AVOD, empowered by deep learning, embodies a transformative approach to object detection in autonomous driving. Its ability to leverage diverse image information and achieve exceptional accuracy paves the way for safer and more reliable autonomous vehicles. Continued research addressing the remaining challenges will be crucial in realizing this technology's full potential and shaping the future of transportation.

2.1 Problem Statement

The Bird's Eye View (BEV) is used to detect objects, which is then projected into three different views: the BEV, the Front View (FV) from LIDAR, and RGB images. Regional features from each view are combined using ROI pooling and a deep fusion network. Together, these fused features predict object classes and perform oriented 2D box regression. MV3D [6], which exclusively employs BEV to provide candidate locations, detects autos and truck well but not pedestrians or cyclists. AVOD [7] improves MV3D by proposing regional candidates using BEV and RGB images.

We may fuse this model to create region recommendations for a two-stage object detector and incorporate the regional multimodal properties of each scheme. This research aims to address these challenges by developing a novel object detection framework that combines two prominent methodologies: Fast Region-based Convolutional Neural Networks (Fast R-CNN) and the You Only Look Once (YOLOv8) model, augmented with Practical Swarm Optimization (PSO). The integration of the Feature Pyramid Network (FPN) and the Region Proposal Network (RPN) from Fast R-CNN is intended to improve feature extraction and region proposal generation, while the YOLO model offers a powerful blend of speed and accuracy.

3. METHODOLOGY

In this section, we propose and hybrid approach towards advancing object detection in the realm of autonomous driving through the AVOD framework, employing cutting-edge deep learning techniques. Our methodology encompasses several crucial stages, including data preprocessing, model architecture design, training procedures, and evaluation metrics. By leveraging the AVOD paradigm, we aim to enhance the accuracy, efficiency, and robustness of object detection systems, thereby facilitating safer and more reliable autonomous navigation on the roads. We delineate each step in our methodology, elucidating the rationale behind our decisions and detailing the experimental setup for empirical validation.

3.1 Dataset Description

The KITTI dataset [40] is a widely used benchmark for object detection, tracking, and scene understanding tasks, particularly in the context of autonomous driving. It consists of approximately 1960 urban traffic scenes captured by camera images and lidar-based point clouds. These scenes provide rich sensor data, including camera images and corresponding point clouds. The dataset is meticulously annotated with bounding boxes around various traffic participants such as cars, cyclists, and pedestrians.

Table 2. Statistics of the KITTI dataset used for evaluation

Dataset	Scenes	Annotations		
		Car	Cycle	Pedestrians
Training	700	300	250	150
Validation	760	300	200	260
Test	500	150	150	200

These annotations are crucial for training and evaluating algorithms that aim to detect and track objects in urban environments accurately. To facilitate comprehensive evaluation, the KITTI dataset defines three difficulty levels: easy, moderate, and hard based on several factors. These factors include the size of the object in the camera image, the degree of occlusion, and the truncation ratio. This hierarchical categorization allows researchers to assess algorithm performance across different levels of complexity, reflecting real-world challenges encountered in urban driving scenarios. For training and validation purposes, the annotated scenes are divided into distinct splits, namely the training split and the validation split (See Table 2). This division ensures that images from the training and validation sets are drawn from disjoint sequences, preventing data leakage and ensuring the integrity of the evaluation process.

3.2 Workflow of Proposed Model

The proposed workflow offers a comprehensive and synergistic approach to object detection, leveraging the complementary strengths of RPN, FPN, Fast R-CNN, and YOLO models, enhanced further through hybridization and optimization techniques. Particle Swarm Optimization (PSO) iteratively updates a swarm of candidate solutions to search for the optimal configuration of parameters, enhancing the model accuracy and efficiency. Through extensive experimentation and evaluation, we demonstrate the effectiveness of our approach in achieving state-of-the-art results in various object detection tasks, paving the way for more robust and accurate systems in real-world applications.

3.2.1 Aggregative View of Object Detection (AVOD)

AVOD processes RGB and BEV features separately using two identical feature extractors. The second stage detection network refines dimensions, direction estimates, and classification of these feature maps. The network structure is shown in Figure 2. Figure 2 shows the two encoder and decoder feature extractors that AVOD has suggested. The encoder has reduced the number of channels in CNN in half and chopped the network in half at the Conv4 layer, among other changes. So, the encoder receives a M×N×D image or a panoramic view as input and produces a (M/8) × (N/8) × D feature map. Pedestrians typically take up 0.8×0.6m in the KITTI datasets and 8×6 pixels in the BEV (with a resolution of 0.1m). The resulting feature map has just one pixel after the encoder down-samples it eight times. Prior to merging the two coders using a 3×3 convolution, a bottom-up decoder is created to return the sample size from the encoder's feature map to its original input size. Then, the feature maps of the two encoders are joined using a conv-transpose. Shared between RPN and the second stage detection network, the final feature maps acquired by the encoder and decoder are known for their excellent resolution and representativeness. For a deep convolution network, the most abstract feature information is stored in the network's deepest feature map. Consequently, the suggested skip feature pyramid model can fuse the detail information from low-level feature maps with the scale information from high-level feature maps, and vice versa. This allows for effective utilization of the scale information between feature layers.

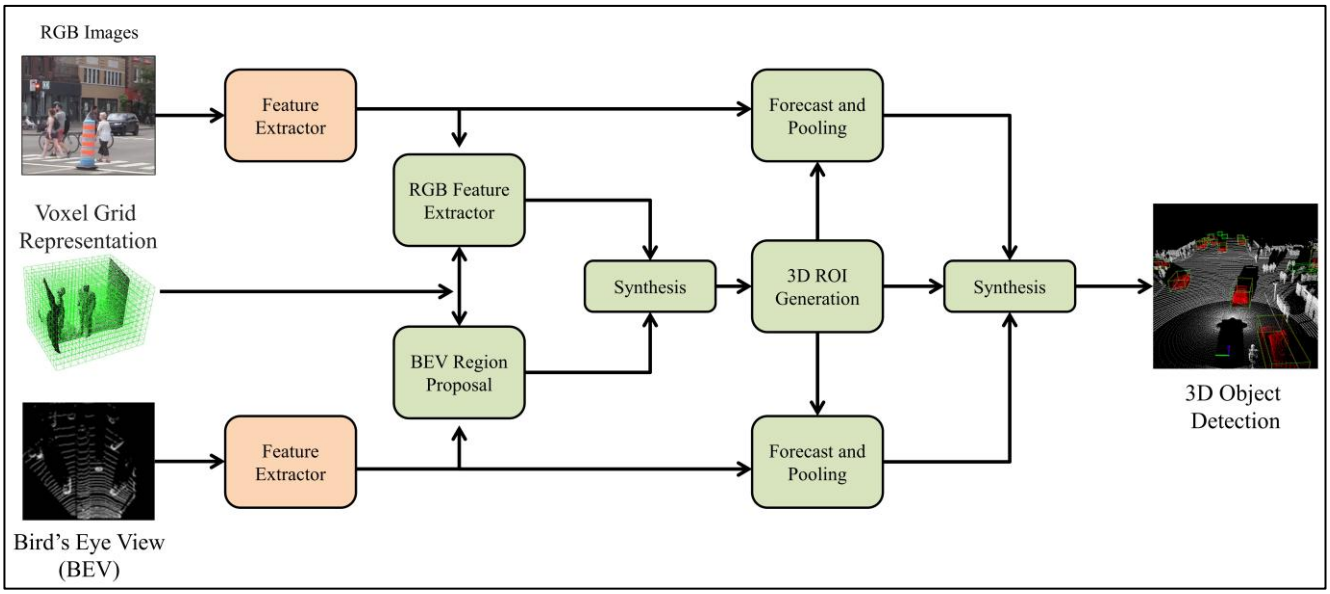


Figure 2. Architecture of AVOD

Using the basic network's high-level feature map selection, convolution kernels with 3×3 channel sizes of 32, 64, and 128 are employed to hide each high-level feature map. This method's goal is to standardize the number of channels used for fusion calculations by converting feature maps from several feature layers to a common number. Once the number of channels in each layer has been unified, the feature map of neighbouring feature layers is adopted using a 2×2 deconvolution procedure. Once the upper sampling is complete, all of the feature layers are equal in size. Afterwards, feature maps that are not contiguous are sampled using a 4×4 deconvolution size.

3.2.2 Faster Region-Based Convolutional Neural Network

The Faster R-CNN structure, depicted in Figure 3, comprises both the Region Proposal Network (RPN) and the Convolutional Neural Network (CNN). The heart of Faster R-CNN, the RPN, generates high-quality region proposals as a fully convolutional network. The convolutional layer creates feature maps from any size image. The RPN generates region recommendations from this feature map using a customized convolutional layer to produce higher-dimensional features. ROI pooling makes these higher-dimensional characteristics and region proposals homogeneous in size. To calculate coordinates and scores, ROI pooling output is sent into fully-connected box-regression and box-classification layers.

Faster R-CNN extracts target proposal regions using RPN: A tiny network glides over the final convolutional layer's feature map, fully connecting to the same-sized window on each slide. One 512-length vector per sliding window. This lower-dimensional vector is supplied into two sibling fully-connected box-regression and box-classification layers. The classification layer generates 4k coordinates for each box at each sliding-window location, whereas the regression layer outputs 2k scores to estimate the object to be foreground or background. Each n × n sliding window has an anchor centre and can forecast k proposals for each location. The scale, aspect ratio, and anchor of each of these k suggestions determines its reference box. To select the voxel, the voxel grid representation is categorized into two types: positive and negative. This classification is based on the IoU overlap ratio between the ground-truth box and the anchor box, which serves as the classification index.

$$IoU = \frac{S_{anchorbox} \cap S_{groundtruth}}{S_{anchorbox} \cup S_{groundtruth}} \quad (1)$$

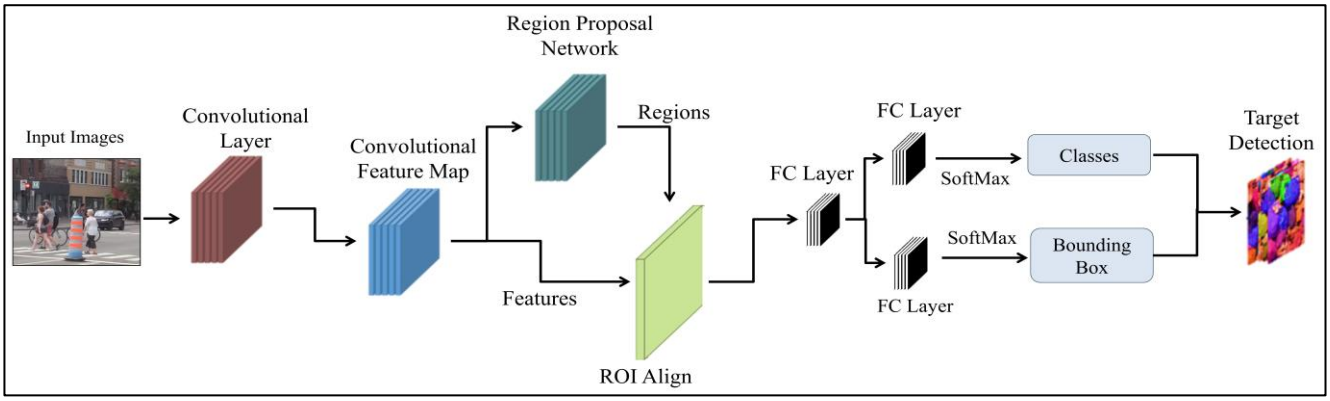


Figure 3. Fast R-CNN for object detection

The following are the rules for classification. **Rule 1:** an anchor is considered a positive sample if its IoU is larger than that of the ground-truth box. Voxels are considered positive samples according to **Rule 2:** if their IoU values are greater than 0.7. Using rule 2 usually yields sufficient positive samples. However, in the most severe instances, rule 1 is still applied. Take anchors as an example; their IoUs are all smaller than 0.7. **Rule 3:** an anchor is considered a negative sample if its IoU is less than 0.3. **Rule 4:** The rest anchors aren't used for training purposes and aren't positive or negative samples. The **Rule 5:** is that any anchor box that goes beyond the edge of the image is likewise removed.

3.2.3 Input Preprocessing and Feature Extraction

The images captured by the RGB camera are normalized by deducting the average RGB value from the dataset used for training. We employ the FR-CNN architecture [27] for feature extraction in images, making the same adjustments as in [25], namely half the number of filters in each convolutional layer, omitting the fifth convolutional stage, and finishing off the fourth stage without a max-pooling layer. The final set of 128 feature maps is 8 times more compact in every direction. Bilinear up sampling four times is used to get feature maps with a better resolution. The lidar point cloud images are cropped to reside in the volume $\triangle$X = [−40m, 40m], $\triangle$Y = [−1m, 3m], $\triangle$Z = [0m, 70m] from the dataset. The volume is partitioned into equally sized voxels of size (s_x, s_y, s_z) = (0.2m, 0.4m, 0.2m). The voxel input is processed by a Feature Pyramid Network, as shown in Figure 3. It consists of grouping, random sampling and stacked voxel feature encoding (VFE) layers. Each point $p_i = (x_i, y_i, z_i)$ in each voxel v forms an input vector $(x_i, y_i, z_i, \tilde{x}_v, \tilde{z}_v)$ with point reflectance r_i and voxel mean coordinates $(\tilde{x}_v, \tilde{z}_v)$. Before max-pooling and concatenation, each VFE layer learns a locally aggregated feature from an input fully connected layer. Two VFE layers are stacked, and we randomly sample 45 points per voxel. Each voxel gets a feature vector with 32 dimensions from the first VFE layer. With the second set of VFE layers, each voxel produces a feature vector with 128 dimensions. Three 3D convolutional layers conv3D (c, k, s, p) with output channels c, kernel size k, stride $s = (s_x, s_y, s_z)$ and padding $p = (p_x, p_y, p_z)$ aggregate voxel-wise features to obtain an expanding receptive field to capture more context information.

In the process of integrating YOLO into a Fast R-CNN for object detection, several key steps are involved to effectively leverage the capabilities of both the backbone architecture and the object detection framework. Firstly, the input image undergoes forward propagation through the CNN network. CNN, a deep convolutional neural network architecture, is adept at extracting hierarchical features from images. These features capture various levels of abstraction, ranging from low-level edge and texture information to high-level semantic concepts. During this feature extraction stage, convolutional feature maps are generated from one of the intermediate layers of CNN, often chosen based on a balance between computational efficiency and representational capacity. For instance, the output of the last convolutional layer, such as conv 5×3, is commonly used due to its rich semantic information. Subsequently, these convolutional feature maps serve as the input to the Region Proposal Network (RPN). The RPN analyses the features extracted by CNN to propose candidate regions, or bounding boxes, likely to contain objects of interest. By examining spatial patterns and semantic cues within the feature maps, the RPN predicts bounding box coordinates and assigns abjectness scores to each candidate region, indicating the likelihood of containing an object. The process of Region of Interest (ROI) pooling follows the production of region proposals. This is where ROI pooling or a comparable process, like ROI align, is used to extract feature vectors of a specified size for each area proposed from the convolutional feature maps. Subsequent layers can process regions of interest consistently, regardless of their spatial extents, because this technique guarantees spatial alignment and facilitates efficient feature extraction. Last but not least, fully connected layers are used for object classification and regression using the pooled feature vectors. These layers refine the bounding box coordinates for each suggested region and use the extracted characteristics to forecast the class labels. Through the integration of Fast R-CNN spatial and semantic data with the YOLO framework, the model is able to precisely locate and categorize objects in the input image.

3.2.4 YOLO Architecture

The essence of advancements in object detection methodologies, YOLOv8 spearheads a comprehensive strategy for object detection that seamlessly combines effectiveness with precision. At its core lies a sophisticated CNN architecture meticulously crafted to tackle the challenge of detecting objects within images. The network architecture of YOLOv8 is a testament to its ingenuity, comprising a series of convolutional layers followed by detection layers. These layers collectively serve as feature extractors, extracting hierarchical representations from input images with unparalleled efficiency. Unlike its predecessors, YOLOv8 adopts a unified prediction strategy, where bounding boxes and class probabilities are predicted directly from the entire image in a single forward pass through the network. This approach offers a holistic view of the image, enabling YOLOv8 to capture global context and dependencies crucial for accurate object detection. To make YOLOv8 work, you need to use anchor boxes, which are a collection of preconfigured bounding boxes with exact width-to-height metrics. The network is able to pinpoint objects with remarkable accuracy because to these anchor boxes, which act as reference templates for predicting bounding boxes of different sizes and forms. The creation of an all-encompassing loss function that incorporates several aspects of object detection is fundamental to the YOLOv8 training process.

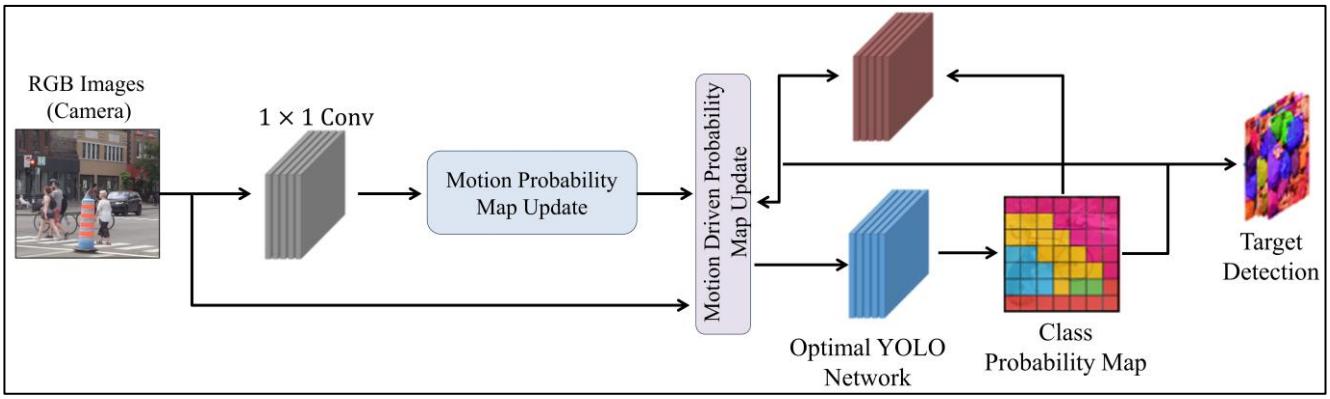


Figure 4. YOLO architecture for object detection

Localization, loss, confidence, and class loss are the three primary parts of this loss function. Discrepancies in predicting bounding box coordinates are penalized by localization loss, whereas errors in predicting the presence or absence of objects are penalized by confidence loss. Conversely, class loss is a penalty for incorrectly categorizing objects. Together, these components guide the network towards learning accurate object localization and classification. During training, YOLOv8 learns to minimize the total loss function across a vast dataset of labelled images using PSO optimizer with backpropagation. Through iterative optimization, the network refines its parameters to predict precise bounding boxes and class probabilities for objects. In the realm of inference, YOLOv8 shines with its real-time performance, swiftly applying the trained model to unseen images to predict bounding boxes and class probabilities. With a single forward pass through the network, YOLOv8 delivers rapid and accurate object detection results, making it a preferred choice for various computer vision applications.

3.3 Proposed Hybrid Model RCNN-YOLO for Object Detection

Object detection using Region-based Convolutional Neural Networks (R-CNN) with YOLO (You Only Look Once) is a sophisticated approach to identifying and localizing objects within images. This technique involves several interconnected steps, each crucial for the model's effectiveness and accuracy. The heart of the object detection system lies in its architecture, which typically comprises several components. Firstly, a powerful backbone network, such as VGG, ResNet, or Darknet, is employed to extract relevant features from input images. These features serve as the foundation for subsequent processing. A Region-based Convolutional Neural Network (RCNN) is used for further refinement and classification after a Region Proposal Network (RPN) has produced prospective object bounding boxes. Another option is that YOLO divides the input image into a grid, which allows it to directly predict bounding boxes and class probabilities. At the outset, the weights of the backbone network are trained using data from a massive image classification dataset, such as ImageNet.

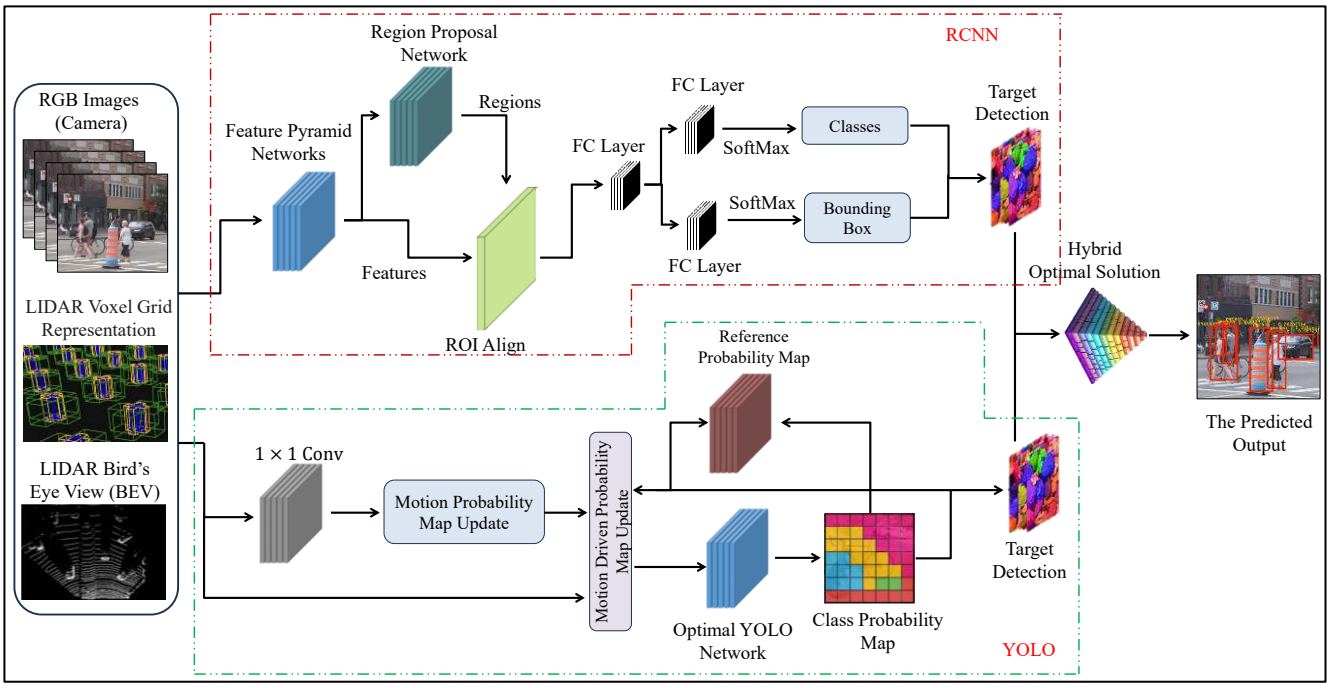


Figure 5. Hybrid RCNN-YOLO architecture for object detection.

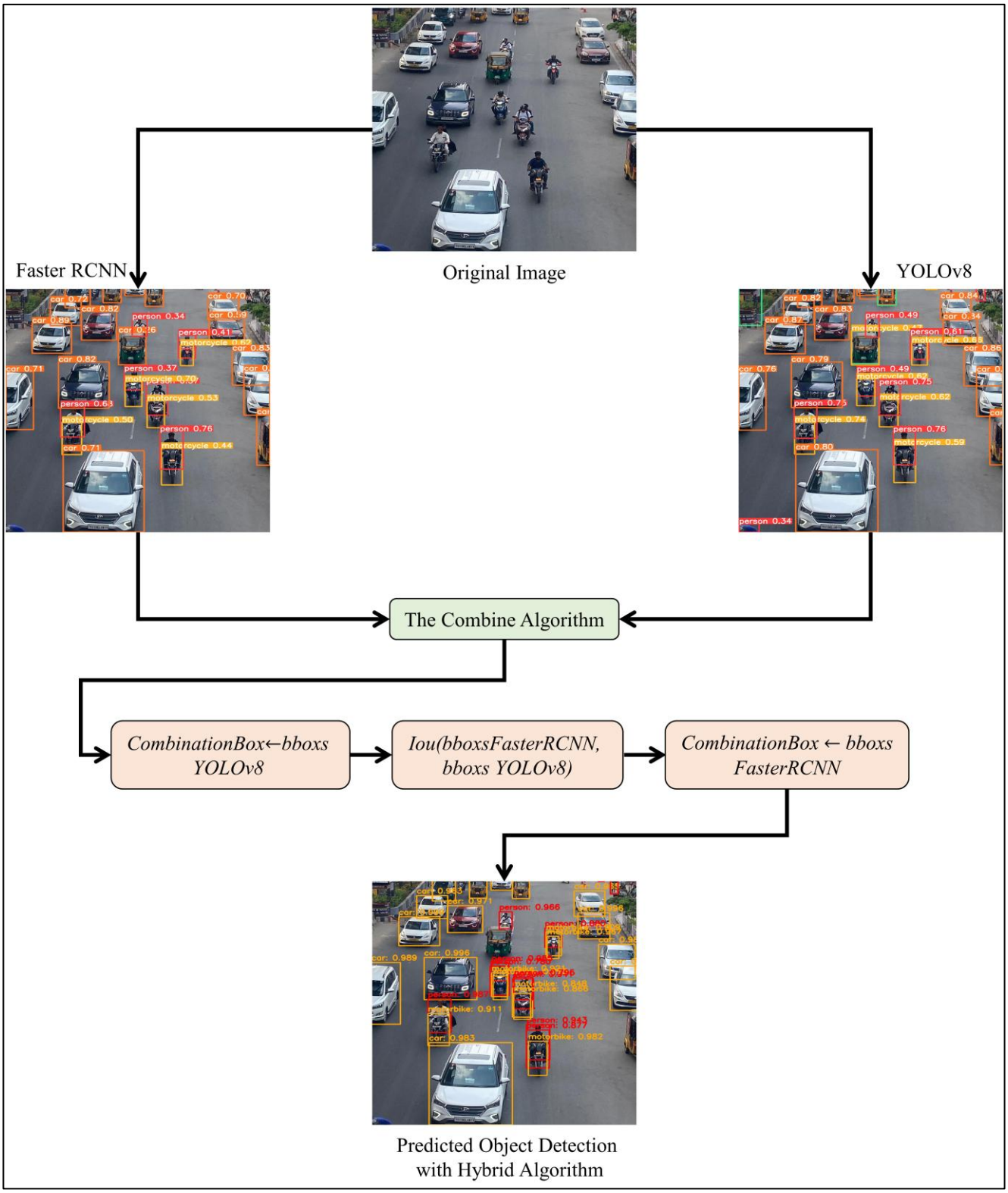


Figure 6. Our combination method, the figure illustrates how our method can combine the predicted bounding boxes from both considered object detectors.

Next, bounding box regression loss and SoftMax cross-entropy loss are minimized to achieve fine-tuning. This composite loss function penalizes both localization and classification errors. In the context of optimizing Fast R-CNN with YOLO, Swarm Optimization techniques, such as PSO, play a valuable role. These techniques excel in navigating complex optimization landscapes and can assist in various aspects of model development. PSO can efficiently search the hyperparameter space, facilitating the selection of optimal learning rates, weight decays, and other crucial parameters. Additionally, it can guide the exploration of network architectures, aiding in the selection of configurations that maximize detection performance.

Algorithm 1: Hybrid RCNN-YOLOv8

```
function (bboxs Faster RCNN; bboxsYOLOv8)
    Combination Box ← []
      for i = 0 to len (bboxs YOLOv8) do
        Combination Box   bboxs YOLOv8
      end for
    for i = 0 to len (bboxs Faster RCNN) do
        count ← 0
    for j = 0 to len (bboxs YOLOv8) do
        result ← IOU (bboxs FRCNN; bboxs YOLOv8)
        if result >= 0:5 then
          count ← count + 1
        end if
    end for
    if count < 1 then
        Combination Box ← bboxs Faster RCNN
    end if
  end for
  return Combination Box
end function
```

Moreover, Swarm Optimization can assist in feature selection, regularization, and other aspects critical for model robustness and generalization.

3.3.1 Particle Swarm Optimization (PSO) for Training the Neural Network Weights and Hyperparameter Tuning

PSO plays a pivotal role in refining the integration of Fast R-CNN and YOLOv8 for enhanced object detection capabilities. This optimization technique is employed to systematically fine-tune various critical parameters within both models. For Fast R-CNN, PSO optimizes the configuration of the FPN, which enhances feature extraction across multiple scales by adjusting the number of feature maps and receptive field sizes. Concurrently, PSO optimizes the RPN parameters in Fast R-CNN, refining aspects such as anchor sizes, aspect ratios, and the number of anchors per location to improve the quality and accuracy of region proposals. In the context of YOLOv8, PSO fine-tunes parameters such as anchor box dimensions, loss function coefficients, and network architecture specifics, aiming to achieve a balance between speed and detection accuracy. This iterative optimization process not only aims to boost performance metrics on established datasets like KITTI and custom datasets but also highlights PSO's effectiveness in achieving optimized configurations that maximize the overall object detection efficiency and accuracy of the integrated Faster R-CNN and YOLOv8 models. By integrating PSO into the methodology, the approach demonstrates a synergistic enhancement where metaheuristic optimization complements deep learning techniques, ultimately advancing state-of-the-art capabilities in object detection tasks. Each particle tends to travel randomly while heading towards the current global best g^* and its own historical best location x_i^*. PSO changes a particle's best position when it finds one. An optimal position exists for all n particles at any moment t throughout iterations. The goal is to find the global best from all the existing best solutions until the objective no longer improves or after a set number of iterations.

Algorithm 1: Particle Swarm Optimization

Data: Object function $f(x)$
Result: Best or optimal solution
Initial location $\boldsymbol{x_i}$ and velocity v_i of n particles
Find $\boldsymbol{g}^*$ from min $\{f(x_1), \ldots, f(x_n)\}$ at $t = 0$;
while criterion **do**
  for all particles and all d dimensions do
    Generate new velocity v_i^{t+1} using equation (8.1);
    Calculate new locations $x_i^{t+1} = x_i^t + v_i^{t+1}$;
    Evaluate the objective function at new locations x_i^{t+1};
    Find the current best for each particle x_i^*;
  end
  Find the current global best $\boldsymbol{g}^*$
  Update $t = t + 1$ (pseudo time or iteration counter);
end
Output the final results $\boldsymbol{x_i^*}$ and $\boldsymbol{g}^*$

Let x_i and v_i be the position vector and velocity for particle respectively. The new velocity vector is determined by the following formula:

$$v_i^{t+1} = v_i^t + \alpha\epsilon_1[g^* - x_i^t] + \beta\epsilon_2[x_i^{*(t)} - x_i^t] \quad (2)$$

Here, $\in_1$ and $\in_2$ are two random vectors with each entry taking values between 0 and 1. The parameters α and β are the learning parameters or acceleration constants, typically set to around $\alpha \approx \beta \approx 2$. The initial positions of all particles should be distributed relatively uniformly to ensure they sample across most regions, which is crucial for multimodal problems. The initial velocity of a particle can be set to zero, that is, $v_i^{t=0} = 0$. The new position is then updated by:

$$x_i^{t+1} = x_i^t + v_i^{t+1}\Delta t \quad (3)$$

The time increment is represented by Δt. Given that PSO is iterative and uses a discrete integer time counter, we can provide $\Delta t = 1$ for any implementation. The value of v_i is not limited to any particular range but is typically limited to a value between range $[0, v_{max}]$. One enhancement to the basic PSO method is the use of the inertia function $\theta_{(t)}$ to replace v_i^t with $\theta(t)v_i^t$ expressed in Eq. (4).

$$v_i^{t+1} = \theta v_i^t + \alpha\epsilon_1[g^* - x_i^t] + \beta\epsilon_2[x_i^{*(k)} - x_i^t] \quad (4)$$

where θ varies from 0 to 1. In the most basic scenario, the inertia function can be assumed to be constant, usually with θ ranging from approximately $\theta \approx 0.5 \sim 0.9$. Adding a virtual mass to the system stabilizes the particles' motion, which should lead to faster algorithm convergence.

4. RESULTS AND DISCUSSION

In this section, we provide the results of the comparison with the hybrid technique that was proposed on the KITTI benchmark as well as the in-house dataset. The purpose of

this experiment is to test the efficacy of a hybrid algorithm with PSO that combines the YOLO v8 detector with the Faster R-CNN detector in a Python environment. In order for YOLO v8 and Faster R-CNN to work, the 1920×1200 resolution images are initially downsized to 255×255. We do quantitative evaluation on easy, medium, and hard difficulty classification. The computational efficiency and memory needs of the proposed network are performed using Intel TRX 4080 8GB GPU, our Feature Pyramid fusion architecture processes frames in 0:1 second with 20ms pre-processing.

4.1 Feature Extractor

In the suggested design, each input view makes use of a separate but complementary feature extractor architecture. Two parts, an encoder and a decoder, make up the full resolution feature extractor, as seen in Figure 7. Based on FRCNN, the encoder makes a few tweaks, such as halving the number of channels and severing the network at the conv-4 layer. Hence, a M×N×D image or BEV map can be fed into the encoder, which then generates an $\frac{M}{8} \times \frac{N}{8} \times D^*$ feature map. Although it has 8 times less resolution than the input, F. Here, F has a high level of representational capability. According to the KITTI statistics, the typical footprint of a pedestrian in a BEV is 0.8:0.6 meters.

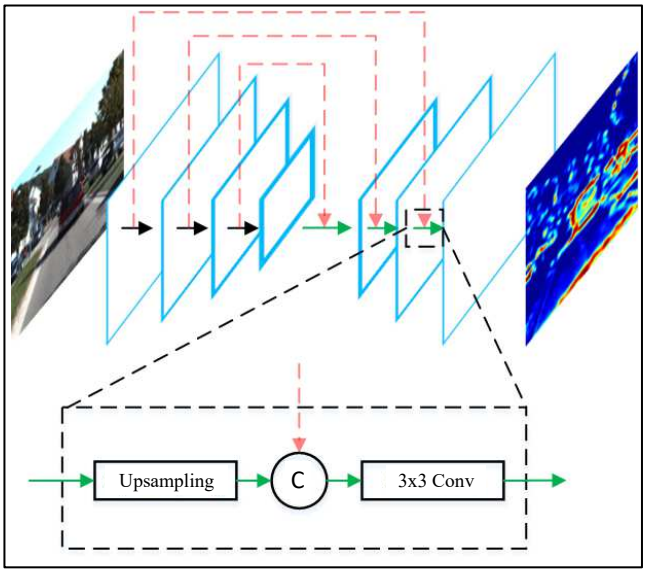


Figure 7. Our proposed image branch high-resolution feature extractor architecture. Red arrows carry feature maps from encoder to decoder. A learnt up-sampling layer fuses, concatenates, and mixes via a convolutional layer at every stage of the decoder, culminating in a full-resolution feature map at the last layer.

This corresponds to a 0:1 meter resolution section in a BEV map that is 8 × 6 pixels wide. Even before taking into account the increase in receptive field caused by convolution, these small classes occupy less than one pixel in the final feature map following 8× down-sampling. The FPN [5] served as an inspiration for our bottom-up decoder, which is able to learn to up-sample the feature map to the original input size while maintaining a constant run time speed. By accepting F, the encoder's output, as input, the decoder creates a new M×N×$\widetilde{D}$ feature map. Figure 7 shows the decoder's process, which includes processing the input with a conv-transpose operation, combining the encoder's matching feature map, and then merging the two with a 3 × 3 convolution operation. The second-stage detection network and RPN both make use of the final feature map, which is highly representative and has excellent resolution.

Extracting Feature Crops Via Multiview Crop and Resize Operations: We perform the crop and resize process to get feature crops for each anchor from the view-specific feature maps [18]. Two areas of interest can be extracted from an 2D anchor by superimposing it onto the BEV and image feature maps. Next, feature map crops are extracted from each view using the associated regions. These crops are subsequently bilinearly enlarged to 3 × 3 in order to create feature vectors of identical length. This extraction method outperforms Faster-RCNN's original 3 × 3 convolution by producing feature crops that adhere to the projected anchor's aspect ratio in both perspectives. We test proposed model object detection and proposal generating capabilities on our inhouse dataset as well as the three classes from the KITTI object detection benchmark [1]. We used [4] as a basis to divide the 1960 frames in creating a training set, testing and a validation set. We classify the tasks using easy, medium, and high difficulty levels for evaluation. Specifically, we trained designed networks for automobiles, pedestrians, bicycles, and pedestrians together. The RPN and detection networks are trained end-to-end using mini-batches that contain one image and either 512 or 1024 ROIs. Train the network for 90 iterations using an exponentially decaying PSO optimizer with an initial learning rate of 0.001 and a decay factor of 0.4.

4.2. Qualitative Results

We present some qualitative representative results generated by our proposed hybrid model on the test split of proposed dataset in Figure 6. In order to generate precise bounding boxes, our part-aggregation network takes point cloud inputs and uses them to predict the locations of intra-object components, as depicted in the image. The inhouse test set also allows us to compare our design against VoxelNet[10], F-PointNet[12], and MV3D [14]. Results from the test sets are provided by the evaluation server with the AHS metric being computed. Our solution is twice as fast as F-PointNet and outperforms it on all three classes, even with just the encoder for feature extraction. Compared to F-PointNet, our architecture achieves a 2:10% improvement in performance on hard (very obstructed or distance) examples after using our high-resolution feature extractor, feature pyramid. This surpasses all existing state-of-the-art methods.



Figure 8. Proposed model objection detection

4.3 Ablation Experiment

In the conducted experiments, various hyperparameters were investigated to assess their impact on the proposed model parameters, Average Precision (AP), and Average Heading Similarity (AHS) performance. The baseline network utilized the feature extractor from MV3D along with the hyperparameter settings specified in the original paper. This study aimed to understand the relationship between the origin of the Region Proposal Network's (RPN) input feature vector and the final detection AP by training a network with RGB, LiDAR, and BEV features as model inputs. The experiments evaluated the effects of incorporating an orientation regression output layer and different bounding box encoding schemes on both AP and AHS performance. A high-resolution feature extractor was compared against the MV3D's feature extractor to determine any performance gains. The analysis of input variations for the proposed model, as depicted in Figure 9, demonstrated that integrating features from multiple perspectives at the RPN stage significantly improved AP and AHS for pedestrians and cyclists when compared to using BEV features alone. Specifically, with 1024 proposals, pedestrian AP and AHS increased by 8.3%, and cycling recall improved by 5.7%. Further, the addition of a high-resolution feature extractor reduced the performance gap to 10% for all classes. However, for the car class, using the high-resolution feature extractor or incorporating image features into the RPN did not show any enhancement in AP and AHS over the BEV-only approach. This integrated methodology, which combines Fast R-CNN and YOLOv8 with Particle Swarm Optimization (PSO), exhibited substantial improvements in object detection accuracy and efficiency. The employment of PSO for hyperparameter optimization, along with enhanced feature extraction and region proposal generation, led to superior performance across standard datasets and complex scenarios, such as those encountered in autonomous driving.

The findings highlight the effectiveness of this approach in achieving high accuracy and real-time applicability, making it a robust solution for contemporary object detection challenges. The improvements are particularly noteworthy in complex environments, where the combination of RGB, LiDAR, and BEV inputs, along with advanced feature extraction techniques, plays a crucial role in enhancing detection capabilities. This comprehensive strategy not only boosts performance metrics like AP and AHS but also ensures that the system remains efficient and applicable in real-time scenarios, which is essential for practical applications such as autonomous vehicles. Overall, the ablation study underscores the significant benefits of combining multi-perspective features and advanced optimization techniques in object detection frameworks, paving the way for more accurate and reliable detection systems in dynamic and complex environments.

Table 4. Ablation experiment for object detection.

Method	AP_{Easy}	$AP_{Moderate}$	AP_{Hard}
YOLOv8	79.80	82.30	84.40
FRCNN	89.12	86.12	87.12
CNN-PSO	88.24	89.14	86.24
This Work	**98.33**	**97.12**	**95.13**

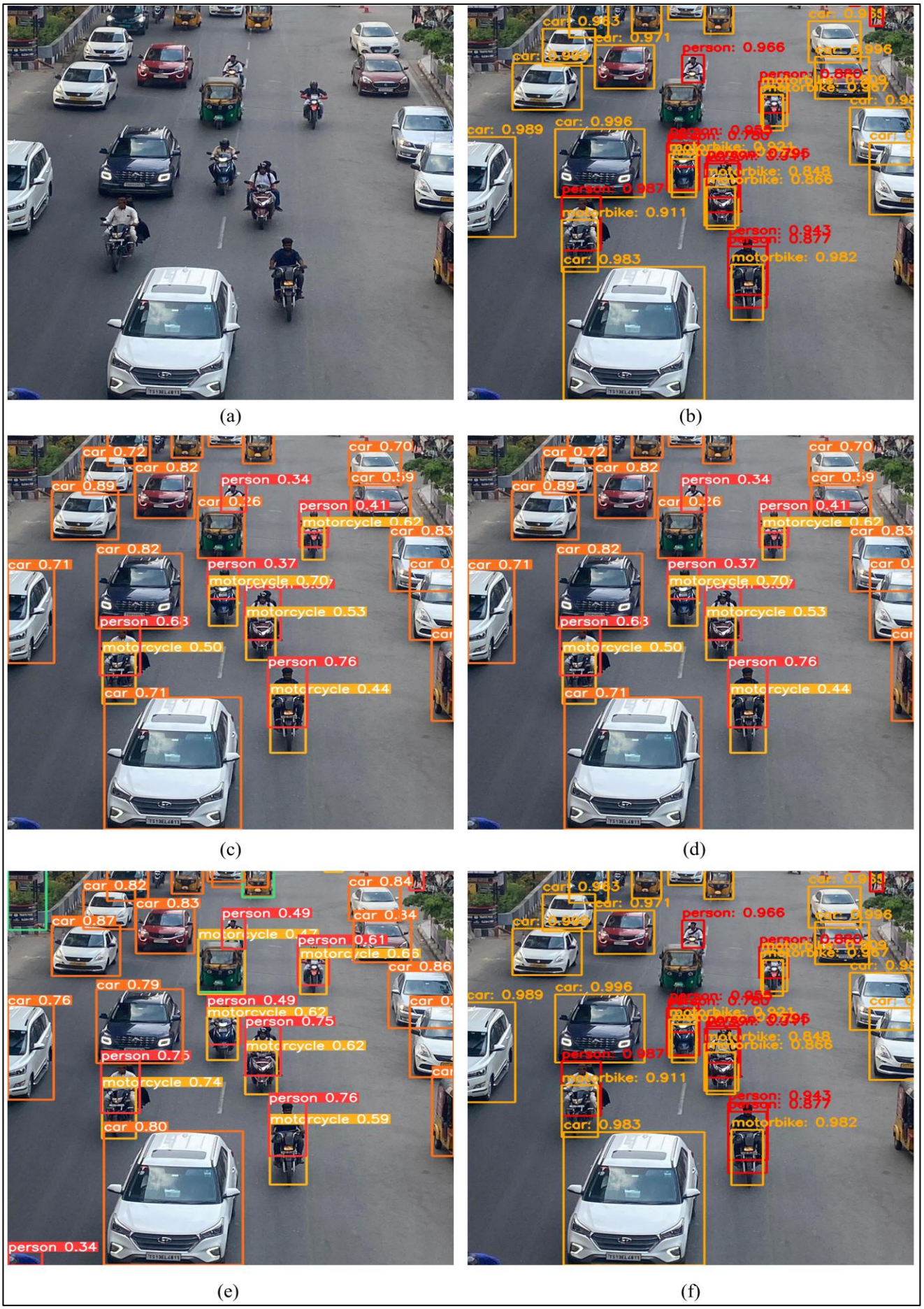


Figure 9. (a) original image, (b) ground truth image, (c) YOLO, (d) Fast R-CNN, (e)PSO-CNN, (f) Proposed Model.

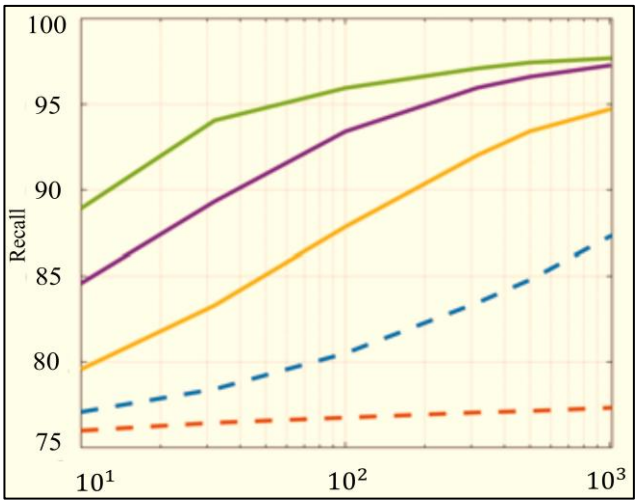


Figure 10. Recall of proposed model represents in green colour.

Using image features at the RPN stage increases pedestrian and cyclist AP by 6:9% and 9:4%, respectively, over BEV just. Figure 9 provides a visual representation of the outcomes of AVOD and MV3D in comparison to the ground truth that was determined by inhouse dataset.

4.4 Quantitative evaluation results of the proposed objection detection

We evaluate our results and compare them to those of state-of-the-art object detection methods. Our proposed model, which improves accuracy in ablation investigations, is our main focus. Table 4 displays the outcomes. Using RGB + LiDAR + BEV, our approach surpasses all multi-sensor methods on the crucial easy, moderate, and hard difficulty level for the car, person, pedestrian and bike classes in the object detection benchmark, as well as all peer-reviewed LiDAR-only methods across all three classes.

Table 4. Performance of different Quantitative evaluation matrices using automatic methods for object detection.

Method	Runtime	Class	Modality	AP			AHS		
				Easy	Mod.	Hard	Easy	Mod.	Hard
MV3D [41]	0.43s	Car	RGB + LiDAR	73.19	62.35	55.12	86.02	76.90	68.49
ContFuse [5]	0.50s		RGB + LiDAR	74.17	65.11	57.73	89.35	79.26	77.39
AVOD-FPN [4]	0.46s		LiDAR only	82.10	70.39	62.19	88.70	84.00	75.33
F-PointNet [6]	0.60s		RGB + LiDAR	74.39	65.78	58.38	86.80	85.44	77.73
VoxelNet [32]	0.34s		LiDAR only	82.14	71.88	66.38	88.53	83.79	77.90
PointRCNN [8]	0.45s		RGB + LiDAR	61.31	44.89	40.23	86.02	76.90	68.49
RPN BEV [10]	0.33s		RGB + LiDAR	33.28	31.51	26.98	89.35	79.26	77.39
Deep3DBox [33]	0.25s		LiDAR only	84.32	76.90	68.49	88.70	84.00	75.33
This Work	**0.13s**		**RGB + LiDAR + BEV**	**98.33**	**97.12**	**95.13**	**97.33**	**98.10**	**97.14**
MV3D [41]	0.43s	Pedestrian	RGB + LiDAR	72.19	66.25	59.32	87.53	87.89	**76.10**
ContFuse [5]	0.50s		RGB + LiDAR	76.27	64.21	54.13	83.12	79.10	70.39
AVOD-FPN [4]	0.46s		LiDAR only	83.10	76.19	68.39	87.12	79.30	77.19
F-PointNet [6]	0.60s		RGB + LiDAR	81.19	79.38	53.18	**88.15**	82.16	80.19
VoxelNet [32]	0.34s		LiDAR only	**87.74**	72.18	65.18	86.10	85.10	79.13
PointRCNN [8]	0.45s		RGB + LiDAR	52.11	49.29	82.13	88.40	**86.14**	80.13
RPN BEV [10]	0.33s		RGB + LiDAR	76.18	65.21	62.88	87.63	88.29	79.30
Deep3DBox [33]	0.25s		LiDAR only	87.12	78.10	69.59	89.12	78.70	69.80
This Work	**0.13s**		**RGB + LiDAR + BEV**	86.34	**83.13**	**85.15**	82.25	85.43	81.34
MV3D [41]	0.43s	Cyclist	RGB + LiDAR	74.19	86.35	65.32	**89.25**	78.16	90.29
ContFuse [5]	0.50s		RGB + LiDAR	76.17	90.21	75.43	88.30	83.90	89.13
AVOD-FPN [4]	0.46s		LiDAR only	84.10	81.29	78.49	86.20	86.34	90.13
F-PointNet [6]	0.60s		RGB + LiDAR	74.29	69.38	85.28	92.13	84.69	80.10
VoxelNet [32]	0.34s		LiDAR only	82.84	79.48	86.68	90.12	79.10	80.19
PointRCNN [8]	0.45s		RGB + LiDAR	52.31	**83.29**	90.23	90.25	80.26	79.19

RPN BEV [10]	0.33s		RGB + LiDAR	80.18	81.12	88.18	89.30	85.10	79.13
Deep3DBox [33]	0.25s		LiDAR only	85.12	77.80	86.29	87.30	89.14	80.63
This Work	0.13s		RGB + LiDAR + BEV	**86.34**	83.13	**87.15**	82.25	**89.43**	**81.34**

For reference, Table 4 shows our car, pedestrian, and cyclist performance. Compared to other state-of-the-art methods our approach enhances car performance while maintaining pedestrian, and cyclist performance. Our algorithm may perform poorly on pedestrians because the sparse point cloud makes it difficult to distinguish pedestrian direction, which hinders part location prediction.

5. CONCLUSION

In conclusion, this paper introduces a novel combination framework designed to address the limitations of current object detectors, particularly when it comes to detecting overlapping and small objects within images. By integrating two renowned detection models, Faster R-CNN and YOLO, our approach demonstrates a notable improvement in the detection of objects, providing more accurate bounding boxes and increasing the overall number of detected objects. The experimental results validate the efficacy of our method, showcasing its potential to enhance the performance of existing detectors significantly. This improvement is particularly beneficial for applications where overlapping objects are prevalent, such as in autonomous driving systems. The ability to detect such objects more reliably can lead to advancements in safety and functionality in these applications. However, it is important to acknowledge that our current method requires further refinement in terms of computation time. Reducing the processing time is crucial for real-time applications and for ensuring the practical usability of the framework.

Looking ahead, there is considerable potential to extend this method to object detection, which could open up new possibilities for dynamic and real-time object detection scenarios. Future work will focus on optimizing computation efficiency and exploring the application of this framework in various real-world environments. Overall, our combination framework represents a promising step forward in the field of object detection, offering a robust solution to some of the existing challenges and paving the way for further research and development.

Ethical Statement(s)

Availability of supporting data

The datasets analyzed during the current study are not publicly available due to individual privacy but are available from the corresponding author on reasonable request.

Competing interests

The authors declare no competing interests.

Funding

No funding was received for conducting this study.

Consent to Publish: Not Applicable

Ethical Approval and accordance: Not Applicable

Consent to Participate declaration: not applicable

References

1. Iqra, Kaisar J. Giri, and Mohammed Javed. "Small object detection in diverse application landscapes: a survey." Multimedia Tools and Applications (2024): 1-36.
2. Kaur, Jaskirat, and Williamjeet Singh. "A systematic review of object detection from images using deep learning." Multimedia Tools and Applications 83.4 (2024): 12253-12338.
3. Tian, Di, Yi Han, and Shu Wang. "Object feedback and feature information retention for small object detection in intelligent transportation scenes." Expert Systems with Applications 238 (2024): 121811.
4. Yu, Xiaoyong, and Xiaoqiang Lu. "Domain adaptation of anchor-free object detection for urban traffic." Neurocomputing (2024): 127477.
5. Zhang, Yan, et al. "DsP-YOLO: An anchor-free network with DsPAN for small object detection of multiscale defects." Expert Systems with Applications 241 (2024): 122669.
6. Wang, Shaobo, et al. "YOLOH: You Only Look One Hourglass for Real-time Object Detection." IEEE Transactions on Image Processing (2024).
7. Wang, Shihao, et al. "Design and Implementation of SSS-Based AUV Autonomous Online Object Detection System." Electronics 13.6 (2024): 1064.
8. Yuan, Maoxun, et al. "Improving RGB-infrared object detection with cascade alignment-guided transformer." Information Fusion 105 (2024): 102246.
9. Kalgaonkar P, El-Sharkawy M. An Improved Lightweight Network Using Attentive Feature Aggregation for Object Detection in Autonomous Driving. Journal of Low Power Electronics and Applications. 2023; 13(3):49.
10. Ghasemieh, Alireza, and Rasha Kashef. "3D Object Detection for Autonomous Driving: Methods, Models, Sensors, Data, and Challenges." Transportation Engineering 8, (2022): 100115.
11. Pang, Y.; Cao, J. Deep Learning in Object Detection. In Deep Learning in Object Detection and Recognition; Jiang, X., Hadid, A., Pang, Y., Granger, E., Feng, X., Eds.; Springer: Singapore, 2019; pp. 19–57.
12. 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, Las Vegas, NV, USA, 27–30, 2016.
13. Jiang, Chenchen, Huazhong Ren, Xin Ye, Jinshun Zhu, Hui Zeng, Yang Nan, Min Sun, Xiang Ren, and Hongtao Huo. "Object Detection from UAV Thermal Infrared Images and Videos Using YOLO Models." International Journal of Applied Earth Observation and Geoinformation 112, (2022): 102912.
14. Kaur, Ravpreet, and Sarbjeet Singh. "A Comprehensive Review of Object Detection with Deep Learning." Digital Signal Processing 132, (2022): 103812.
15. Lin, T.-Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature Pyramid Networks for Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26, 2017.
16. N. Jia, Y. Sun and X. Liu, "TFGNet: Traffic Salient Object Detection Using a Feature Deep Interaction and Guidance Fusion," in IEEE Transactions on Intelligent Transportation Systems, pp. 1-11, 2023.
17. Kamath, Vidya, and A. Renuka. "Deep Learning Based Object Detection for Resource Constrained Devices: Systematic Review, Future Trends and Challenges Ahead." Neurocomputing 531, (2023): 34-60.
18. Sirisha, U., Praveen, S.P., Srinivasu, P.N. et al. Statistical Analysis of Design Aspects of Various YOLO-Based Deep Learning Models for Object Detection. Int J Comput Intell Syst 16, 126 (2023).
19. Y. Cao, C. Li, Y. Peng and H. Ru, "MCS-YOLO: A Multiscale Object Detection Method for Autonomous Driving Road Environment Recognition," in IEEE Access, vol. 11, pp. 22342-22354, 2023.
20. Li Z, Gao Y, Hong Q, Du Y, Serikawa S, Zhang L. Keypoint3D: Keypoint-Based and Anchor-Free 3D Object Detection for Autonomous Driving with Monocular Vision. Remote Sensing. 2023; 15(5):1210.
21. Arnold, E.; Al-Jarrah, O.Y.; Dianati, M.; Fallah, S.; Oxtoby, D.; Mouzakitis, A. A survey on 3D object detection methods for autonomous driving applications. IEEE Trans. Intell. Transp. Syst. 2019, 20, 3782–3795
22. Dhillon, Anamika, and Gyanendra K. Verma. "Convolutional neural network: a review of models, methodologies and applications to object detection." Progress in Artificial Intelligence 9.2 (2020): 85-112.
23. Long, Xiang, et al. "PP-YOLO: An effective and efficient implementation of object detector." arXiv preprint arXiv:2007.12099 (2020).
24. Liu, Li, et al. "Deep learning for generic object detection: A survey." International journal of computer vision 128 (2020): 261-318.
25. Wu, Xiongwei, Doyen Sahoo, and Steven CH Hoi. "Recent advances in deep learning for object detection." Neurocomputing 396 (2020): 39-64.
26. Li, Yanghao, et al. "Exploring plain vision transformer backbones for object detection." European Conference on Computer Vision. Cham: Springer Nature Switzerland, 2022.
27. Ding, Jian, et al. "Object detection in aerial images: A large-scale benchmark and challenges." IEEE transactions on pattern analysis and machine intelligence 44.11 (2021): 7778-7796.
28. Kumar, Ashwani, Zuopeng Justin Zhang, and Hongbo Lyu. "Object detection in real time based on improved single shot multi-box detector algorithm." EURASIP Journal on Wireless Communications and Networking 2020.1 (2020): 204.
29. Gong, Yuqi, et al. "Effective fusion factor in FPN for tiny object detection." Proceedings of the IEEE/CVF winter conference on applications of computer vision. 2021.
30. Shen, Yanyun, et al. "On-board multi-class geospatial object detection based on convolutional neural network for High Resolution Remote Sensing Images." Remote Sensing 15.16 (2023): 3963.
31. Qin, Xuebin, et al. "U2-Net: Going deeper with nested U-structure for salient object detection." Pattern recognition 106 (2020): 107404.
32. Deng, Shengheng, et al. "Vista: Boosting 3d object detection via dual cross-view spatial attention." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022.
33. Yang, Zetong, et al. "3dssd: Point-based 3d single stage object detector." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020.
34. Pang, Su, Daniel Morris, and Hayder Radha. "CLOCs: Camera-LiDAR object candidates' fusion for 3D object detection." 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020.
35. Wu, S., Li, X., & Wang, X. (2020). IoU-aware single-stage object detector for accurate localization. Image and Vision Computing, 97, 103911.
36. Chen, Y., Li, Y., Zhang, X., Sun, J., & Jia, J. (2022). Focal sparse convolutional networks for 3d object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 5428-5437).
37. Razi, A., Chen, X., Li, H., Wang, H., Russo, B., Chen, Y., & Yu, H. (2023). Deep learning serves traffic safety analysis: A forward-looking review. IET Intelligent Transport Systems, 17(1), 22-71.
38. Zou, Z., Chen, K., Shi, Z., Guo, Y., & Ye, J. (2023). Object detection in 20 years: A survey. Proceedings of the IEEE, 111(3), 257-276.
39. Kaur, R., & Singh, S. (2023). A comprehensive review of object detection with deep learning. Digital Signal Processing, 132, 103812.
40. Srivastav, A., & Mandal, S. (2023). Radars for autonomous driving: A review of deep learning methods and challenges. IEEE Access.
41. X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving", in Computer Vision and Pattern Recognition, pp. 6526-6534, 2017.