IET Radar, Sonar & Navigation

WILEY

The Institution of Engineering and Technology

ORIGINAL RESEARCH OPEN ACCESS

DetDSHAP: Explainable Object Detection for Uncrewed and Autonomous Drones With Shapley Values

Maxwell Hogan | Nabil Aouf

Department of Engineering, City, University of London, London, UK

Correspondence: Maxwell Hogan (Maxwell.Hogan@city.ac.uk)

Received: 29 November 2024 | **Revised:** 7 May 2025 | **Accepted:** 21 May 2025

Handling Editor: Hugh Griffiths

Funding: The authors received no specific funding for this work.

Keywords: artificial intelligence | object detection | object recognition

ABSTRACT

Automatic object detection onboard drones is essential for facilitating autonomous operations. The advent of deep learning techniques has significantly enhanced the efficacy of object detection and recognition systems. However, the implementation of deep networks in real-world operational settings for autonomous decision-making presents several challenges. A primary concern is the lack of transparency in deep learning algorithms, which renders their behaviour unreliable to both practitioners and the general public. Additionally, deep networks often require substantial computational resources, which may not be feasible for many compact portable platforms. This paper aims to address these challenges and promote the integration of deep object detectors in drone applications. We present a novel interpretative framework, DetDSHAP, designed to elucidate the predictions generated by the YOLOv5 detector. Furthermore, we propose utilising the contribution scores derived from our explanatory model as an innovative pruning technique for the YOLOv5 network, thereby achieving enhanced performance while minimising computational demands. Lastly, we provide performance evaluations of our approach demonstrating its efficiency across various datasets, including real data collected from drone-mounted cameras and established public benchmark datasets.

1 | Introduction

Autonomous and uncrewed drones have seen a surge in demand in civilian applications such as agriculture [1] and search and rescue [2] and defence applications such as automatic target recognition (ATR) [3]. Object detection is often required to enable a drone to undertake its tasks. Since the adoption of deep neural networks (DNNs), there have been remarkable performance gains in automatic object detection capabilities. Nonetheless, deploying deep networks onboard drones remains challenging due to their computational requirements and their unpredictability when compared to traditional computer vision algorithms.

The integration of algorithms into compact drone platforms presents significant challenges related to memory and processing capabilities, complicating their implementation. Because of constraints in memory and power, achieving real-time performance with deep networks can be particularly challenging, and in certain instances, it may be infeasible to accommodate the entire deep network within the memory limits of specific drone platforms. Network pruning seeks to enhance the efficiency of deep networks by refining the network architecture while striving to minimise any associated decline in performance.

On platforms equipped with adequate embedding hardware (memory), the inherent closed-box characteristics of deep

networks continue to render them unreliable for deployment in real-world applications [4]. Enhanced trust can be fostered by elevating the transparency of these networks. This process resembles the act of unveiling the inner workings, thereby enabling the human agent—whether a designer or an end user—to understand the model's decision-making mechanisms. Explainable artificial intelligence (XAI) concept seeks to enhance transparency by elucidating the rationale behind the predictions generated by deep learning models. Techniques for explainability can be employed to uncover biases within the network [5], assist in debugging a network [6] and offer an alternative approach for evaluating outcomes [7].

In this paper, we propose an innovative framework aimed at explaining the individual bounding boxes predicted by the YOLOv5 deep learning detector [8] based on DeepSHAP [9]. DeepSHAP is capable of attributing a contribution score to each input within a deep neural network and has been applied in multiple domains of computer vision. Notably, it has been utilised in reinforcement learning for the autonomous navigation of unmanned aerial vehicles (UAVs) [10], medical image analysis [11] and image classification [12]. The application of DeepSHAP to automatic object detection remains unestablished. This is primarily due to the necessity for object detectors to provide justifications for both the classification and positioning of potentially multiple bounding box candidates, making the straightforward implementation of DeepSHAP a complex endeavour.

In the application of DeepSHAP, the contribution score is propagated backward from the output layer to the input layer. This process allows for the implicit calculation of each unit's contribution within the network. The resulting scores indicate the importance of each component of the model. By aggregating the SHAP values derived from various samples, it becomes possible to create a map that highlights the areas of the model that hold substantial significance, as well as those that may be considered redundant. One could generalise units to mean weights, whole layers or even modules. In our work, we focus on filter-level pruning in convolutional nets.

Previous studies have introduced explainers specifically designed for deep object detectors, and other literature has suggested the application of XAI as a criterion for pruning in a broader context. However, there is a notable absence of research that has explored the convergence of these two areas. Consequently, this paper aims to contribute to the field by examining this intersection (Figure 1).

A major obstacle to the advancement of XAI is the absence of established benchmarks and a lack of consensus within the academic literature regarding the evaluation of novel explanation techniques. Numerous studies assess the effectiveness of their explanation methods by juxtaposing the generated explanations with human-annotated bounding boxes [4, 13–15]. In this study, we present the wrapping game as a validation method that employs semantic segmentation datasets to assess the discriminative capabilities of our explainer in comparison to the explainer developed by Karasmanoglou et al. based on layer-wise relevance propagation (LRP) [16] for general pruning purposes. Additionally, here the following axiom is surmised: a

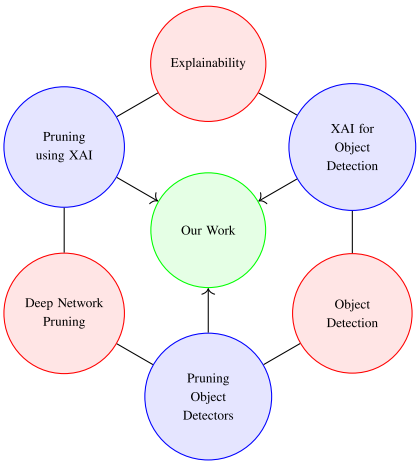


FIGURE 1 | Domain of the present paper. Conceptual landscape of this study. Red circles mark the three broad research strands: Explainability, object detection, and deep-network pruning. Previous works have exploited links in the between these broader area to produce specialised frameworks. The green circle at the centre represents our contribution, which combines all three strands in a single framework.

propagation-based XAI method that offers a robust criterion for pruning must also be allocating causal information correctly. Specifically, this demonstrates that the contribution scores derived from our explanatory framework not only serve as an effective pruning criterion but also validate that our explainer accurately allocates causal information.

We intend to propose a solution for object detection that emphasises explainability and pruning, designed to address the diverse requirements of drones functioning in urban settings. This initiative is motivated by the current deficiency of explainable algorithms capable of achieving state-of-the-art performance across a wide array of datasets [17]. To support our claims, we assess our explainability and pruning techniques using two datasets derived from different drone platforms. Additionally, we include an evaluation on the COCO2017 [18] dataset to facilitate comparisons with future research endeavours.

To meet the operational demands of drones in dense urban environments, an object-detection solution must satisfy two requirements simultaneously: it must be light enough to run in real time on constrained, battery-powered hardware and transparent enough that its decisions can be trusted by a remote operator or safety auditor. Existing work treats these goals separately. Pruning techniques reduce detector size, but the pruning masks are chosen heuristically and give no indication why particular filters are deleted; detector-oriented XAI methods such as Grad-CAM, LRP and RISE reveal influential features, yet they leave the heavy backbone untouched and thus do not relieve the compute bottleneck. To our knowledge no framework integrates an explanation-guided pruning strategy with a detector-specific explainer. This paper fills that gap: we present a joint explainability–pruning approach and evaluate it on two drone datasets together with COCO 2017, demonstrating state-of-the-art accuracy, compactness and interpretability across all three.

Our main contributions in this paper are as follows:

- A new explainable artificial intelligence (XAI) framework has been developed for object detection methodologies, aimed at querying bounding boxes that were either predicted or overlooked by the chosen YOLO-based detector architectures.
- A presentation of our YOLO detector explainer, which has been trained on our proprietary self-driving car dataset as well as the Visdrone public dataset, illustrates the effectiveness of our explainer in accurately attributing the contributions of individual neurons within our network and the corresponding input image.
- An assessment of the wrapping game analysis demonstrates the effectiveness of our deep object detector explainer in distinguishing between different objects.
- A newly developed framework for pruning DNN detectors utilising the contribution metrics provided by DetDSHAP as a criterion. This approach significantly enhances the efficiency of the YOLOv5 detector while maintaining minimal performance degradation.
- A comprehensive quantitative examination of the impact of our pruning methodology on the efficiency and quality of our deep object detection system.

2 | Background and Motivation

This section provides the conceptual background that motivates our method. Section 2.1 reviews recent work on explainable object detection, highlighting what current saliency and relevance approaches can, and cannot, show. Section 2.2 then surveys pruning methods that rely on explanation signals, summarising their strengths and limitations when applied to large detector backbones. Finally, Section 2.3 introduces the SHAP/DeepSHAP framework that underpins our own explainer and explains why it is well suited to serve as a joint explanation–pruning criterion. Together, these three subsections expose the gap our work fills: no existing approach couples a detector-specific explainer with an explanation-driven, filter-level pruning strategy.

2.1 | XAI for Object Detection

Explainers for deep detectors fall into three possible types. Perturbation-based surrogate methods—such as RISE, D-RISE and KernelSHAP—estimate saliency by masking or shuffling the input and fitting a simple surrogate model. Gradient-based single-layer methods (e.g., Grad-CAM) back-propagate first-order derivatives only to the final convolutional block, while propagation-based all-layer methods (notably LRP and its variants) redistribute the detector's output score through every layer. All three types of explainer ultimately produce a saliency map—a heat-map that highlights how strongly each pixel influences the network's prediction [19]. Much of the detector-oriented literature adapts a saliency technique originally developed for classification and repurposes it for bounding-box tasks [4, 14, 16, 20]. The following paragraphs review key examples from each type in turn.

In ref. [13], the authors propose to use Randomised Input Sampling for Explanation (RISE) to provide insights from any type of classifier—even those not based on DNNs. Their method involves creating a surrogate interpretable model to represent a single instance. The surrogate model in this case is a weighted average on a new dataset consisting of perturbed samples. These perturbed samples are given to the closed box model to produce the output probability for the target class—this score is provided as the weighting. RISE was adapted to D-RISE by Petsiuk et al. [14] for use with detection style architectures. Their method is consistent with RISE, except the weights for the weighted average are calculated using a new similarity metric. This similarity metric is meant to measure how different a detected box prediction from a perturbed image is from the original.

This approach is proposed in our previous work in ref. [4] to adapt KernelSHAP [12], which is a method for generating SHAP values and will be explained in detail below, to explain detections made by Uncrewed Aerial Vehicle (UAV) platforms. This kind of explainer also creates a surrogate model, in this case, the surrogate is a linear model. Our approach in that paper and the approach in ref. [14] are completely model agnostic. Moreover, both works provide quantitative evidence that supports their explainers' fidelity which is often omitted by other authors. Nonetheless, they only access the input and the output of the model and thus these approaches cannot be used to probe into the inner workings of the deep detector model.

LRP occupies the third type: it explains a detector's output by propagating relevance scores backwards through every layer, thereby accessing the network's internal structure [21]. Early work applied LRP to Single Shot Detector (SSD) via Contrastive Relevance Propagation (CRP) [22], but SSD has since been surpassed by stronger architectures such as YOLOv5 [23]. Karasmanoglou et al. [16] therefore extended LRP to YOLOv5, producing visually appealing heat-maps; yet, as they themselves note, the fidelity of those maps was not quantified, and LRP's gradient roots make it vulnerable to saturation—a phenomenon in which the explainer is susceptible to allocating important features receive near-zero relevance [24].

In summary, perturbation methods are model-agnostic but blind to internal feature flow, gradient methods discard negative evidence, and current propagation methods suffer from saturation and lack quantitative validation. These limitations leave room for a sign-aware, layer-consistent explainer that can double as a principled pruning signal—the gap our DetDSHAP framework is designed to fill.

2.2 | Pruning by Explaining

This section examines the various pruning methodologies, specifically those that utilise explainability as a criterion for pruning. Explanation-guided pruning methods can be grouped along two axes: the importance metric used to score weights or filters and the scope of pruning: local (applied to one or two dense layers) versus global (applied across the entire backbone). Table 1 contrasts representative papers along these two dimensions. The narrative below will begin with an overview of a

TABLE 1 | Explanation-guided pruning methods classified by importance metric and pruning scope.

Reference	Importance metric	Scope
Yeom et al. [25]	LRP relevance	Global
Sabih et al. [17]	DeepLIFT attribution	Global
Lundström et al. [26]	Integrated gradients	Local (dense layers)

contemporary pruning framework before discussing key examples from each cell of the table.

Pasandi et al. [27] describe pruning as a four-stage cycle: first a baseline network is fully trained; next a pruning objective is set—for example, a target compression ratio, a FLOP budget, or an allowable accuracy loss; redundant weights or filters are then removed according to a chosen importance metric; and, finally, the pruned model is fine-tuned. Competing methods differ chiefly in the metric used to score redundancy, the granularity of removal, and the frequency with which this train–prune–tune loop is repeated.

Yeom et al. [25] introduced the use of LRP as a criterion for model pruning. Their research validates this approach across multiple scenarios, offering valuable insights into the pruning of deep networks, albeit with a focus primarily on classification tasks. The authors demonstrated that the LRP criterion can achieve enhanced compression performance across various datasets in comparison to conventional methods. However, it is important to note that LRP may exhibit tendencies toward over-saturation, which is not ideal for a pruning criterion. Furthermore, this work presents evidence suggesting that LRP may not be appropriate for deep object detection tasks, indicating that DeepSHAP serves as a more effective criterion.

An alternative approach to LRP is proposed by Sabih et al. [17], who advocate for the utilisation of DeepLIFT (Deep Learning Important FeaTures) as a criterion for model pruning. DeepLIFT, developed by Shrikumar et al. [24], addresses the challenge of model saturation that affects LRP and other gradient-based techniques. Like LRP, DeepLIFT analyses the contributions of all neurons within the network and the input features through a backpropagation mechanism. However, the key distinction of DeepLIFT lies in its method of comparing each neuron's activation to a reference activation, thereby assigning contribution scores based on the observed differences.

Establishing a reference point for DeepLIFT poses challenges due to its significant impact on the quality of the generated explanations. For instance, in the case of the MNIST dataset, a black image composed entirely of zeros is utilised as the reference for image analysis, while for CIFAR10, the recommendation is to employ a blurred variant of the image. This particular trait of DeepLIFT somewhat undermines the necessity for the explainer to maintain consistency, potentially leading to failures when it encounters unfamiliar data.

Lundstrom et al. [26] utilise Integrated Gradients as a criterion for pruning. Integrated Gradients [28] can be viewed as an extension of DeepLIFT, with the standard reference being an image of zeros. The target network employed in their study is AlexNet [29], which has been trained on a dataset derived from the Mars Rover [30].

Both, Sabih et al. [17] and Yeom et al. [25] implement their criteria within a global pruning framework. In contrast, Lundstrom [26] restricts pruning to the dense layers of their network, where the majority of parameters for AlexNet are concentrated. Unlike global pruning, local pruning involves applying pruning techniques to specific sections of the model, such as individual layers or uniformly across each layer, which generally results in less effective pruning outcomes [31, 32]. The findings of Lundstrom et al. [26] indicate that critical neurons tend to specialise in particular classes. This specialisation likely explains why Yeom et al. [25] observed that employing positive relevance to determine their criterion yielded the most favourable results.

The primary objective of pruning is typically to facilitate the compression of deep networks for deployment on hardware systems. Nevertheless, the studies discussed in this section do not adequately assess their criteria within a practical context involving portable robotic platforms. Only the work by Lundstrom et al. [26] utilises a dataset obtained from a machine vision platform, yet it employs an outdated architecture. In our research, we illustrate that our approach can successfully prune deep object detectors using realistic data collected from both ground-based and aerial drones.

Because existing work is either local in scope or evaluated only on image classifiers, none of these metrics guarantees both computational savings and detector-specific interpretability for large backbones. Hence our adoption of DetDSHAP as a global, sign-aware criterion for pruning object detectors.

2.3 | SHAP and DeepSHAP

In this section, we provide a concise overview of SHapley Additive exPlanations (SHAP) as introduced by Lundberg [12]. They propose that any explanation for a model's prediction can be conceptualised as a surrogate model. This surrogate model serves as an interpretable approximation of the original model. As illustrated in Equation (1), this surrogate model can be represented as a linear model g. In this expression, z' is a feature vector of size M of a given instance and the weights ϕ, are the contribution of each feature to a given outcome. Features may include elements from a table, pixels from an image, or units within a deep learning network.

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j \tag{1}$$

Lundberg et al. introduce KernelSHAP, which is summarised in Section 2.1. This method is characterised as a perturbation-based technique that creates a dataset indicating the presence or absence of a specific feature z'_j, along with the corresponding predictions made by the network for each combination of

features. The surrogate model utilised in this approach is a linear model that is trained using the dataset produced.

Lundberg [12] successfully utilised DeepLIFT to develop DeepSHAP. They argue that incorporating an input can be interpreted as assigning it its actual value rather than its reference value. Consequently, DeepLIFT can be regarded as an efficient approximation technique for the SHAP values ϕ. In a related study, Chen et al. [9] suggest employing a background distribution to signify the reference value. This is achieved by calculating SHAP values for each baseline within a sample group and averaging the resulting attributions. We adopt this methodology as it addresses the invariance constraint to which DeepLIFT is vulnerable. Furthermore, the authors in ref. [9], demonstrate that this approach yields estimated SHAP values that are more closely aligned with the true SHAP values.

DeepSHAP is applied by He et al. [10] to elucidate the workings of a deep neural network deep neural network DNN designed for autonomous navigation. This approach enables the generation of saliency maps, which highlight the significant features within images captured by a 2D camera that contribute to the decision-making process of the deep network. Additionally, the authors explore the impact of various units within their network on the overall output.

3 | Methodology

This section outlines the primary principles of this paper. Initially, we present our explanatory model designed to generate SHAP values, which elucidate the causal information associated with individual bounding boxes generated by architectures based on the YOLOv5 detector. Subsequently, we utilise this explanatory model to develop an innovative pruning framework aimed at optimising our trained models.

3.1 | DeepSHAP for Object Detection (DetDSHAP)

In this sub-section, we present our method for model explanation. When provided with an image I, a deep detection network f, and a target bounding box T, the explainer generates a SHAP map that illustrates the relationship between I and T as interpreted by f. Notably, the bounding box T is not restricted to those predicted by the model; it can also be the groundtruth bounding box or any hypothetical box specified by the developer. This flexibility significantly enhances the developer's ability to investigate and analyse their network.

Algorithm 1 provides a comprehensive overview of the DetDSHAP algorithm, which is structured into three primary phases. The initial phase involves executing a forward pass through the detection network, during which the forward activations are recorded. Specifically, for any layer that is an activation function (e.g., ReLU, sigmoid or tanh), the algorithm saves both the pre-activation input and the post-activation output. This recording is essential for the later stages of backpropagation when attributing contributions. The second phase entails the initialisation of the forward pass output to eliminate information that is irrelevant to T, a process that is elaborated upon in this section. The concluding phase, which results in the generation of the SHAP map, consists of a backward pass that utilises the backpropagation rule set of DeepSHAP. Additionally, we enhance this framework with our proprietary backpropagation rule, which will be detailed later in this section.

In the initialisation phase, following the forward pass, the predictions used for computing the initial contribution ϕ^i are those generated directly by the deep network under investigation. Notably, these predictions are not randomly generated or externally selected; rather, they reflect the network's own output when processing the provided inputs. In our experiments (outlined in Section 4.2), the entire training set is employed to generate these predictions. This comprehensive approach ensures that ϕ^i is derived from a full representation of the input distribution, thereby eliminating potential biases or omissions that could result from random sampling. Consequently, this methodology reinforces the robustness of the subsequent attribution analysis.

ALGORITHM 1 | DetDSHAP Framework

Input: Fully trained deep network F, Target bounding box T, Image I.
Output: SHAP Map ϕ
Step 1: Forward Pass
$x_i = I$
for layer in F **do**
  $x_i = layer(x_{i-1})$
  if layer is type activation **then**
    $save(x_{i-1}, x_i)$
  end if
end for
Step 2: Initialise prediction based on T
$\phi^i = Initialise(x_i, T)$
Step 3: Backwards pass
for layer in $reversed(F)$ **do**
  $\phi^i = apply\_rule\left(layer, \phi^i\right)$
end for
$\phi = \phi^i$ **return** ϕ

Object detectors differ from image classifiers in that they must identify and localise one or more objects within an image, rather than merely assigning a class score. In the context of YOLOv5, the task T encompasses a class score c, a bounding box represented as $b = [x, y, w, h]$, and an objectness score o. To refine the prediction P generated by the function f, it is necessary to preprocess this information to eliminate any irrelevant data, which is categorised as attribution noise. This preprocessing phase is referred to as the initialisation step, resulting in the initial contribution denoted as ϕ^i. Depending on whether the detection task involves a single class or multiple classes, we employ two distinct methodologies.

In the context of a single-class detector, the primary objective is to ascertain the elements that influence the predicted bounding box. To achieve this, we begin by initialising P through the selection of boxes that exhibit a high degree of similarity to the

target box. The similarity between the target box T_{box} and the predicted box P_{box} is quantified through a specific calculation of Intersect over Union (IoU) Equation (2).

$$IoU(T_{box}, B_{box}) = \frac{Area\ of\ Overlap\ of(T_{box}, B_{box})}{Area\ of\ Union\ of(T_{box}, B_{box})} \tag{2}$$

The initial contribution of each box is denoted as ϕ_j^i in Equation (3). In this notation, i refers to the layer index and j indicates the filter index. In practice, for layers where the output represents predicted boxes—as is the case at the output layer—each filter corresponds directly to a specific predicted box. Hence, ϕ_j^i can be interpreted both as the contribution of the *jth* filter at layer i and, equivalently in this scenario, as the contribution of a specific predicted box.

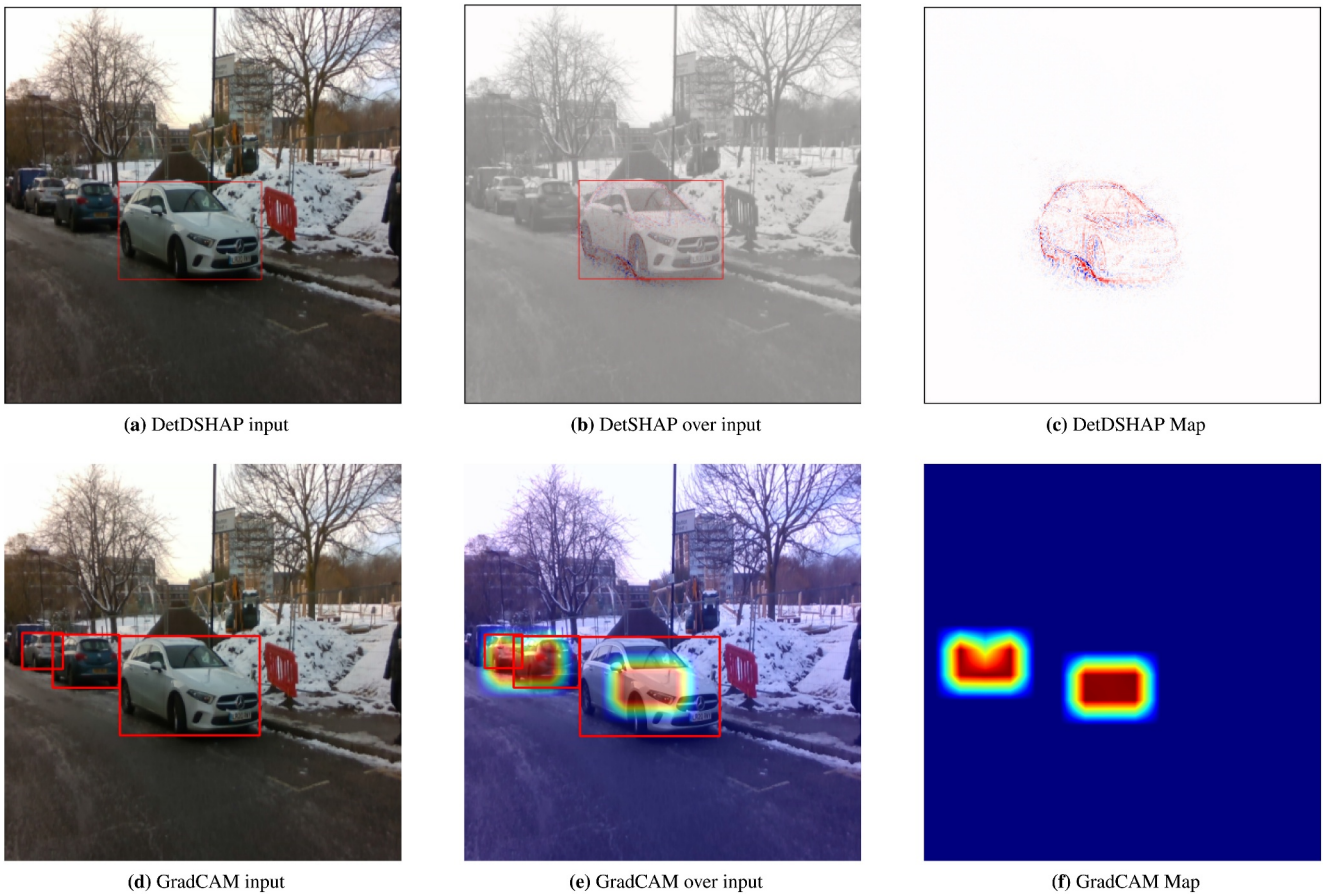
Equation (3) assigns a value to ϕ_j^i, based on the Intersection over Union (IoU) between the target bounding box T_{box} and the predicted bounding box $P_{(j,box)}$. The threshold parameter *thr* ranging from 0 to 1. Our research indicates that a threshold value of $thr = 0.7$ is effective in the majority of scenarios. Subsequently, the initial contribution for P is determined through Equation (4), where M represents the total number of predictions. The resulting SHAP values illustrate the units that significantly impact the localisation of T as exemplified in Figure 2.

Figure 2 contrasts DetDSHAP with a Grad-CAM implementation adapted from [20]. Grad-CAM is applied to the car class-score channel of YOLOv5: because three valid car detections are present (Figure 3d), the explainer is run once per box and the resulting heat maps are averaged. Owing to the ReLU used in Grad-CAM, negative evidence is discarded, and the spatial resolution is limited by the 32 × *stride* of the final feature map. DetDSHAP, by contrast, conditions on a single target box (the foreground vehicle) and propagates SHAP relevance through every layer while retaining both positive and negative contributions.

The visual difference is clear. Grad-CAM produces a broad, low-resolution blob centred on the bonnet and windscreen (Figure 3e,f). DetDSHAP yields a far crisper map (Figure 3b,c): high-relevance pixels trace the bonnet edges, grille and headlights in red, whereas specular snow patches that mislead the detector appear in blue. These properties make DetDSHAP the more informative and object-specific explainer.

$$\phi_j^i = \begin{cases} P_j, & IoU\left(T_{box}, P_{(j,box)}\right) \geq thr \\ 0, & otherwise \end{cases} \tag{3}$$

$$\phi^i = \sum_{j=0}^{M} \phi_j^i \tag{4}$$



(a) DetDSHAP input **(b)** DetSHAP over input **(c)** DetDSHAP Map

(d) GradCAM input **(e)** GradCAM over input **(f)** GradCAM Map

FIGURE 2 | This figure illustrates a comparison of the DetDSHAP map generated using our approach with a saliency map generated from a GradCAM-based framework that was proposed in ref. [20]. Both utilise an image in our proprietary dataset as input.

In the context of a multi-class problem, our methodology yields SHAP values that indicate the units significantly impacting the class scores. Initially, we implement the procedures utilised in a single-class approach to facilitate the localisation of the SHAP computation. Subsequently, we assign a value of one to both the objectness score and the score corresponding to our target class. The SHAP values obtained reflect units that not only play a crucial role in the localisation of T but also in the classification of T.

Figure 3 illustrates an instance of the SHAP maps produced through this methodology. This particular example demonstrates the explainer's ability to differentiate between a rider and their motorcycle. In Figure 3c, the map pertains to the "motorcycle" class, highlighting that the focus of attention is primarily on the vehicle. Conversely, in Figure 3d, which corresponds to the "person" class, it is evident that the silhouette of the rider plays a more significant role in the detection process.

After the initialisation phase, it is possible to perform backpropagation utilising the rules established by DeepSHAP. In the course of backpropagation, the influence of each unit in the network is assessed. Should the user focus solely on the input's contribution to the prediction, they have the option to omit this data to save memory. Nevertheless, this information can be leveraged to illustrate the attention of specific layers as demonstrated in Figure 4.

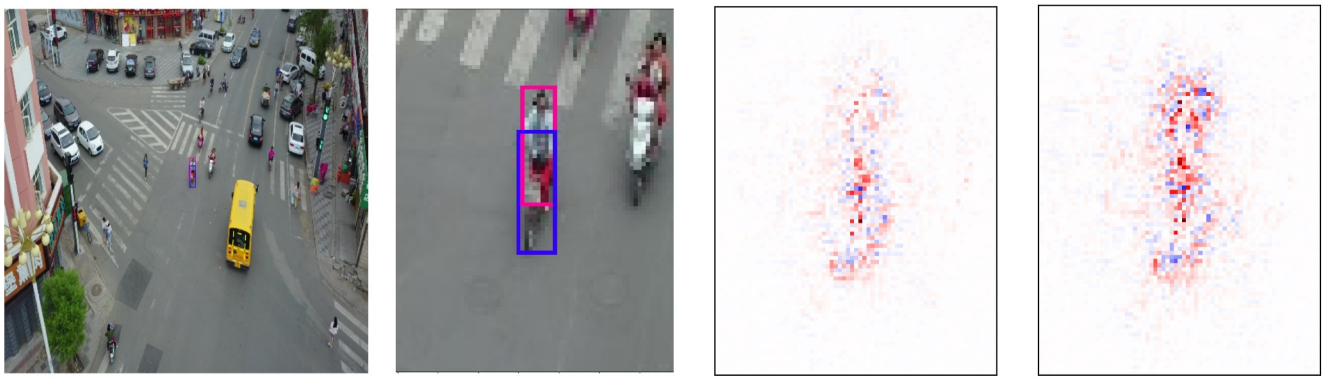


FIGURE 3 | An example of multi-class explanation for Yolov5 trained on Visdrone2021. (a) Shows the image with the target bounding boxes of a person and a motorcycle. (b) Shows the area of interest around the bounding box, (c) The SHAP map generated for the motorcycle, and (d) map for the rider.

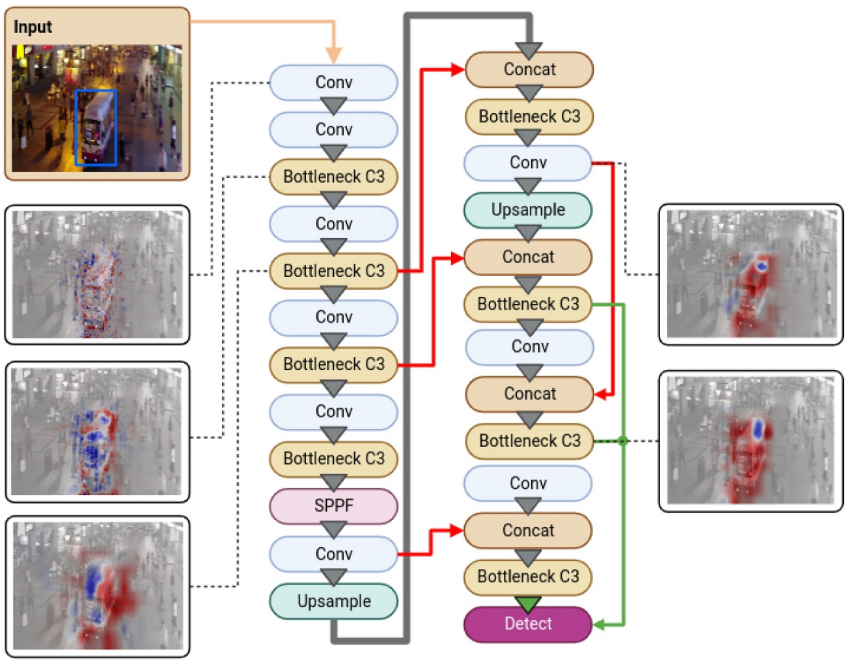


FIGURE 4 | Red arrows indicate feature-aggregation skip routes, the black frame marks the main feed-forward path, and the green arrows link lower-level features to the detection head. The six insets show DetDSHAP maps extracted after successive layers. Feature extraction is performed by repeated Bottleneck-C3 blocks and a SPPF (Spatial Pyramid Pooling–Fast) module, both introduced in the original YOLOv5 design [8].

In conventional DeepSHAP, each nonlinear module registers a backward hook that approximates the local gradient by comparing the module's output activations on the actual input x and a background input x_0. Specifically, let $y(x)$ be the module's output at activation x, and define $\Delta y = y(x) - y(x_0)$, $\Delta x = x - x_0$. The hook then scales the incoming relevance ϕ_{in} by the ratio $\Delta y / \Delta x$:

$$\phi_{\text{out}} = \frac{\Delta y}{\Delta x}\, \phi_{\text{in}}. \tag{5}$$

However, when $|\Delta x|$ is very small this ratio becomes numerically unstable (or undefined), which we found leads to exploding relevance values and invalid saliency maps on YOLO-style networks.

To address this, we implement a custom backward hook for the SiLU activation that falls back to the exact analytic derivative whenever $|\Delta x|$ is below a small threshold $\varepsilon = 10^{-6}$. Recall that:

$$\text{SiLU}(x) = x\, \sigma(x), \quad \text{where} \quad \sigma(x) = \frac{1}{1 + e^{-x}} \tag{6}$$

is the sigmoid function. The true derivative of SiLU with respect to x is

$$\frac{d}{dx}\, \text{SiLU}(x) = \sigma(x) + x\, \sigma'(x) = \sigma(x)\big[1 + x\,(1 - \sigma(x))\big], \tag{7}$$

where

$$\sigma'(x) = \sigma(x)\,(1 - \sigma(x)). \tag{8}$$

In our hook, we compute a scaling factor

$$g(x) = \begin{cases} \dfrac{\Delta y}{\Delta x}, & \text{if } |\Delta x| \geq \varepsilon, \\ \sigma(x)[1 + x\,(1 - \sigma(x))], & \text{if } |\Delta x| < \varepsilon, \end{cases} \tag{9}$$

and then propagate relevance as

$$\phi_{\text{out}} = g(x)\, \phi_{\text{in}}. \tag{10}$$

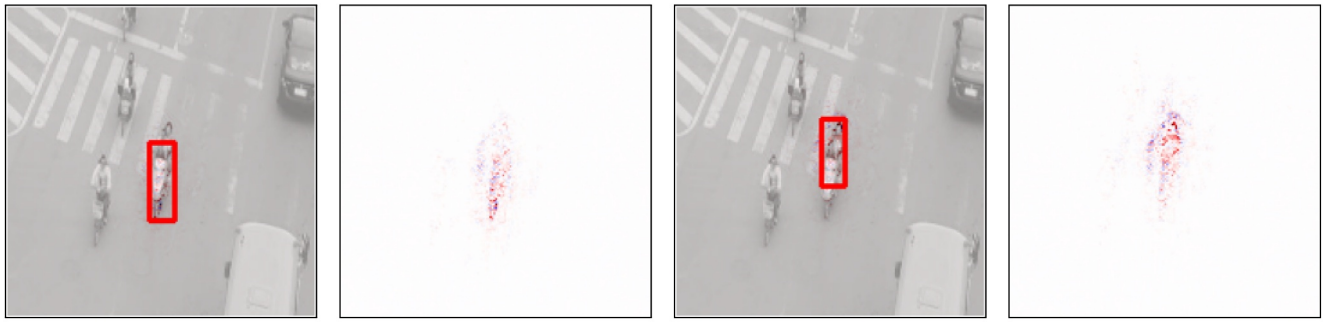
By switching to the analytic SiLU derivative from Equation (7) in regions where $|\Delta x|$ is too small, we avoid division-by-zero and ensure that relevance propagation remains stable. This modification makes DetDSHAP robust enough to generate valid saliency maps on both YOLOv5 and YOLOv11 object detectors.

YOLO-based detection networks generate predictions of offsets relative to predefined anchor boxes, which may be either positive or negative. In instances where an offset is negative, it is essential to assign a higher value of ϕ to the weights that influence the negativity of that offset. To address this, we suggest implementing a specific rule that is applied to the final convolutional layers preceding the detection layer, as illustrated by the green arrows in Figure 4.

The rule under consideration is articulated in Equation (11), drawing inspiration from the LRP-*ab* rule as outlined by Sebastian et al. [33]. In this equation, the term $a_j \cdot w_{jk}$ represents the degree to which filter j (in the i^{th} layer) influences the relevance of filter k (in the $(i+1)^{\text{th}}$ layer); here, a denotes the activations obtained during the forward pass and w the network's learnt weights. The expression $a_j \cdot w_{jk}^{+}$ corresponds to the positive contribution, while $a_j \cdot w_{jk}^{-}$ indicates the negative contribution. The denominator is essential for adhering to the conservation principle established by LRP. Depending on the sign of the activation from the preceding layer (a_j), the incoming contribution (ϕ_k) is adjusted by either the positive or negative relevance.

$$\phi_j^i = \begin{cases} \displaystyle\sum_k \frac{a_j\, w_{jk}^{+}}{\sum_{j'} a_{j'}\, w_{j'k}^{+}}\, \phi_k, & \text{if } a_k \geq 0, \\ \displaystyle\sum_k \frac{a_j\, w_{jk}^{-}}{\sum_{j'} a_{j'}\, w_{j'k}^{-}}\, \phi_k, & \text{if } a_k < 0, \end{cases} \tag{11}$$

The DetDSHAP explainer is compatible with newer YOLO releases such as YOLOv11. To demonstrate this, we generated saliency maps from two YOLOv11 models; one fine-tuned on Visdrone2021 and one on our single-class vehicle detection dataset. The aforementioned maps were generated using the same target boxes shown in Figures 2 and 3. In Figure 5, the DetDSHAP map for YOLOv11 exhibits a noticeably crisper separation between rider and motorcycle: the red positive-relevance regions around the bike's contours no longer bleed



(a) (b) (c) (d)

FIGURE 5 | An example of multi-class explanation for Yolov11 trained on Visdrone2021. The DetDSHAP Maps shown here are generated for the same targets shown in 3b. (a) and (b) illustrate the map generated for the object "Motor", (c) and (d) illustrate the map for the "Person".

into the rider's seat, and the rider's silhouette is more sharply confined. Likewise, in Figure 6, the YOLOv11 vehicle map shows even higher fidelity to the car's grille and headlight features than its YOLOv5 counterpart. These enhancements are expected as different weight sets will learn slightly different feature filters, yet they confirm that DetDSHAP consistently produces high-resolution, object-specific relevance maps across detector versions.

3.2 | DetDSHAP Pruning Framework

In this subsection, we present a framework for globally pruning individual filters in the YOLOv5 detector. For context, Figure 7 shows a generalised overview of the pruning process of DNNs, and Algorithm 2 presents a step-by-step breakdown of our proposed framework. Our approach emphasises the global pruning of individual filters within the network. Throughout the pruning procedure, we systematically reduce the model by a specified quantity (r) until a designated pruning objective, denoted as O, is achieved. Each iteration in this process is termed a pruning step, during which filters are evaluated and ranked based on our established criterion, leading to the removal of those with the lowest rankings. Following each pruning step, the model undergoes fine-tuning to recover any potential decline in performance.

Before each pruning step, let b be a batch of $|b|$ images. For the nth image $x_n \in b$, we collect all of its filter-level SHAP values into the matrix ϕ_n (defined in Equation (12)). Here, I is the total layers and J_i is total filters in layer i. We then aggregate these per-image matrices over the batch by summing absolute values element-wise. Φ_b is the batch-importance matrix whose entries $(\Phi_b)_j^i$ (Equation (13)) give the total contribution of filter j in layer i across all images in b. Filters are ranked by these values and the r lowest are pruned.

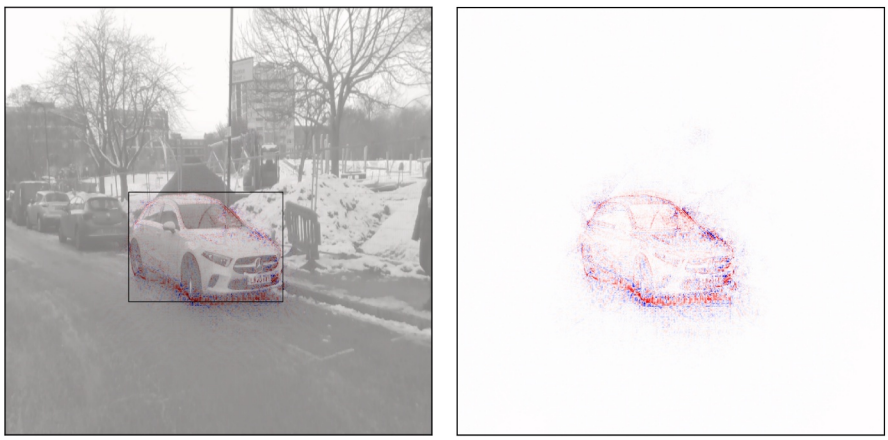
$$\Phi_n = \left[\phi_j^i(x_n)\right]_{\substack{i=1,\ldots,I \\ j=1,\ldots,J_i}} \tag{12}$$

$$(\Phi_b)_j^i = \sum_{n=1}^{|b|} \left|\phi_j^i(x_n)\right| \tag{13}$$

ALGORITHM 2 | Pruning Framework

Input Fully trained deep network F, training data X, pruning objective O, magnitude of pruning step r.
Output Optimised deep network F
  For batch b in X **do**
    Step 1: for each image $x_n \in b$, compute the matrix Φ_n as in Equation (12).
    For x_n in b **do**
      compute SHAP ϕ_n values w.r.t x_n for F
    end for
    Step 2: form Φ_b via Equation (13).
    Step 3: Remove r lowest contributing filters in F with smallest ϕ_b
    Step 4: fine-tuning and check objective
    fine-tune F and X
    if O is True **then**
      break
    end if
  end for
  return F

In accordance with our pruning criterion, we employ the contribution scores derived from our DetDSHAP explainer, as detailed in Section 3.1. Consequently, at the initiation of the pruning process, we compute the SHAP values for a selected batch of samples (b) from the dataset. Each sample consists of an image paired with its corresponding label, which encompasses the 2D coordinates and class labels for all objects present in the image. Although this label typically reflects the model's prediction, our methodology allows us to utilise the ground-truth label for the calculation of SHAP values. This capability enhances our framework, enabling the pruning process to be based on the groundtruth rather than the model's predictions, which is a common limitation in conventional pruning methods.



(a) (b)

FIGURE 6 | Illustrated here is the DetDSHAP map generated for the target vehicle shown in Figure 2. The map was produced to interpret the prediction made by YOLOv11. In (a), the map is overlaid on the input image; in (b), the plain map is shown.

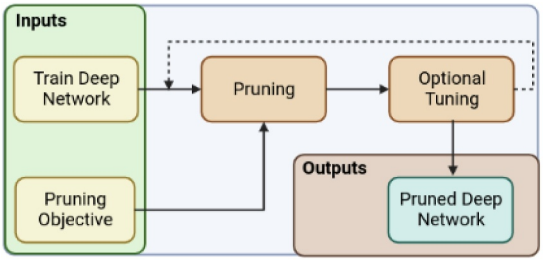


FIGURE 7 | Generalised pruning framework [27].

After generating the SHAP values for each image within a batch, the subsequent task involves determining the rankings of the filters. Research indicates that neurons exhibit a tendency to specialise, suggesting that the negative SHAP values associated with one instance may also be relevant to a nearby instance. Consequently, filters are ranked according to the magnitude of their contributions, with lower rankings assigned to those filters whose SHAP scores are nearest to zero. The ultimate filter rank, denoted as ϕ_b, is computed by aggregating the absolute SHAP values $|\phi_i|$ from each individual sample i.

Once the DetDSHAP ranking identifies filter j in layer i for removal, our PyTorch implementation performs two steps: (1) it rebuilds the convolutional module at layer i with its j-th output channel excised (shifting all higher channels down by one), and (2) it mirrors that change in layer $i + 1$ by deleting the corresponding j-th input channel to keep the tensor shapes aligned. Because modern detectors like YOLOv5 use nonlinear architectures—for example skip-connections through Bottleneck-C3 modules constructed in tight loops—simply removing channels can break the graph. To handle this, we first generate an explicit layer-connectivity graph that records exactly which convolutional modules feed into each other (including those inside the C3 and SPPF blocks). Our custom pruning framework then uses that graph to propagate each channel removal across all affected modules.

The batch-aggregated scores in Equation (13) have an immediate interpretation: given the additive property of SHAP values $(\Phi_b)_j^i$ quantifies how much the network's bounding-box prediction would change, on average, if filter j at layer i were removed. Filters whose contribution oscillates near zero across the data distribution are therefore redundant both spatially (they seldom activate) and semantically (they seldom help or hinder any detection), so pruning them should preserve mAP. Conversely, filters that consistently carry large positive or negative relevance are retained. DetDSHAP ranking is thus data-driven, sign-aware, and box-specific—three characteristics absent from magnitude-only or gradient-only pruning heuristics described in Section 3.1. This intuition underpins the empirical results reported in Section 4.2.

4 | Experimental Results

In this section, we detail the results of the performance analysis performed on our DetDSHAP-based explainer and pruning frameworks. Table 2 illustrates the models employed in each experiment, along with the corresponding datasets on which they were trained. For the Visdrone dataset, which presents significant challenges, we utilised YOLOv5l. In contrast, our self-driving dataset required only YOLOv5s. Additionally, for the COCO2007 dataset, we implemented a pretrained YOLOv5m sourced from [8] to ensure reproducibility. Figure 8 presents samples of each of the dataset employed in this paper.

We initially employ our recently developed wrapping game to assess the discriminatory effectiveness of our DetDSHAP explainer in comparison to the LRP-based method proposed by Karasmanoglou et al. [16]. For the purposes of this specific experiment, we exclusively utilise YOLOv5s which has been trained on our self-driving car dataset in order to reduce the computational expenses associated with our preliminary analysis. The analysis is not exteded to YOLOv11 given that the baseline method of Karasmanoglou et al. does not currently support the v11 codebase.

The subsequent sub-sections examine the efficacy of our DetDSHAP-based pruning framework. Initially, we explore the trends in pruning performance, illustrating the impact of increasing pruning magnitude on the quality and efficiency of the pruned deep model. For comparative analysis, we also assess the effectiveness of the pruning using the LRP-based explainer as described in ref. [16]. We choose three pruned networks and subject them to additional training, subsequently comparing their detection performance with that of their unpruned equivalents.

We conclude with an examination of our deep learning-based detector networks following the pruning process. The networks utilised in this study are all based on the YOLOv5 architecture and have been trained on our proprietary self-driving car dataset, in addition to two publicly accessible datasets: Visdrone [35] and COCO2017 [18]. The efficiency of the deep network is assessed through two distinct metrics. The first metric involves the count of network parameters, which serves to evaluate the extent of compression achieved by the pruning framework. The second metric pertains to the number of Floating point OPerations per Second (FLOPs) necessary for conducting an inference. We illustrate the enhancements in both metrics and present the final results as detailed in Equation (14). Here, "*Score*" refers interchangeably to (i) the total number of network parameters or (ii) the number of floating-point operations (FLOPs) required per inference. The "*Score*" may also refer to the qualitative metrics discussed below.

TABLE 2 | This table shows the models and datasets used in our experimental evaluation.

Model	Dataset	Experiments			
		Wrapping game	Pruning trends	Post pruning performance	Pruning versus design
YOLOv5s	Our dataset [34]	✓	✓	✓	
YOLOv5l	Visdrone		✓	✓	
YOLOv5m	COCO2017		✓	✓	✓

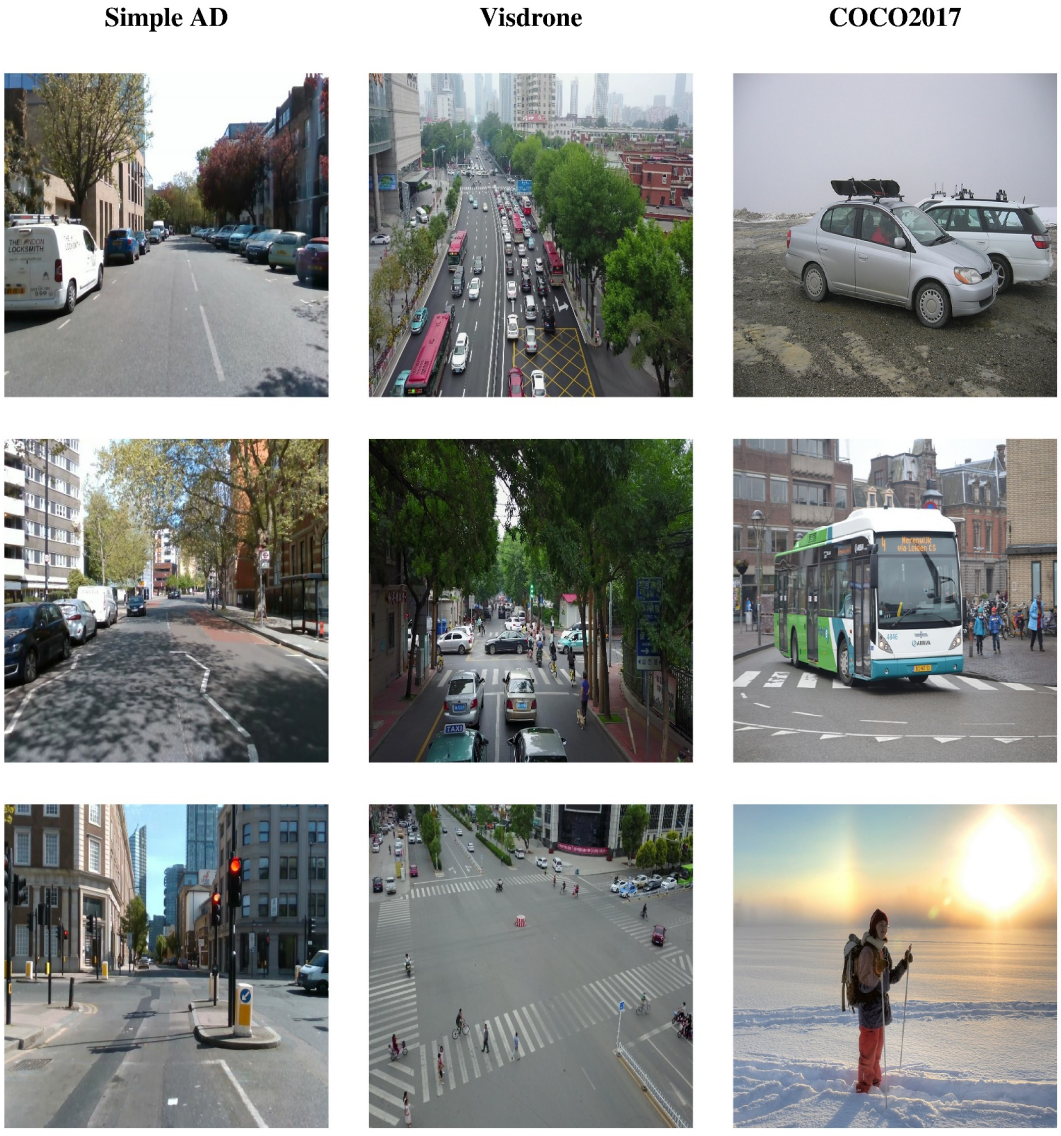


FIGURE 8 | Examples from the three datasets used in the experiments: Simple AD (left), Visdrone (centre), and COCO2017 (right). The COCO2017 images show objects of interest for this study.

$$improvement = 1 - \frac{Score\ on\ pruned\ Model}{Score\ on\ original\ Model} \tag{14}$$

In our research, the efficacy of our deep networks is assessed based on their detection capabilities. To quantitatively evaluate the performance of our pruned networks, we employed four widely recognised metrics: MAP@0.5, MAP@0.5–0.95, Average Recall (AR), and F1 Score. Precision is determined using the formula presented in Equation (15), where a detection is classified as a true positive if the classification is accurate and the IoU between the predicted bounding box and the groundtruth exceeds a specified threshold. Specifically, a true positive is identified when there is an IoU of 50% between the predicted box and the actual groundtruth. The precision is subsequently averaged across all classes to derive the mean average precision (MAP). In the second metric, MAP@0.5–0.95 follows a similar methodology but considers a range of IoU values between 50%

and 95%. The IoU has been previously defined in Equation (2). Additionally, we report the average recall, which is calculated as the ratio of true positives to the total number of groundtruth boxes. The final metric, the F1 score, is computed as outlined in Equation (16), incorporating both recall and precision into its evaluation.

$$precision = \frac{\sum TP}{\sum (TP + FP)} \tag{15}$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{16}$$

4.1 | The Wrapping Game

In this subsection, we present our results from assessing the discriminatory performance of our DetDSHAP-based explainer alongside the LRP-based explainer developed by Karasmanoglou et al. [16], which we designate as YOLO-LRP. To facilitate this evaluation, we introduce a novel methodology termed the wrapping game. Our methodology draws inspiration from the pointing game proposed by Zhang et al. [15], which serves as a tool for examining the discriminatory effectiveness of explainers in image deep network classifiers. In their framework, the most prominent pixel, identified as the highest contributor from a given attention map, is extracted. Subsequently, it is assessed whether this pixel resides within a human-annotated groundtruth represented by a 2D bounding box. A hit is recorded if the pixel is contained within the groundtruth; otherwise, it is classified as a miss. The overall score is calculated by dividing the number of hits by the total of hits and misses. Our analysis of the wrapping game for both explainers is illustrated in Figures 9 and 10.

The pointing game inherently lacks the necessity for a comprehensive explanation to elucidate the complete context of the object in question. Consequently, it can only assess discrimination capabilities to a somewhat restricted extent. In our proposed evaluation framework, we create a mask derived from the explanation map, which facilitates the measurement of the IoU against a groundtruth of instance segmentation. We implement our wrapping game analysis utilising the YOLOv5s model, trained on our self-driving dataset.

The creation of this mask occurs in two distinct phases. Initially, we compute the contribution of each pixel within the image. Subsequently, we employ Sklansky's Algorithm [36] to process all pixels that exceed a specified contribution threshold, resulting in the formation of a polygon.

The criterion employed for selecting pixels in Sklansky's Algorithm is defined as a percentage of the most relevant pixel. Figure 9 illustrates the variation in the average IoU as the threshold value is incremented. The IoU score rises until it attains a maximum, corresponding to the mask nearing its optimal configuration, after which it starts to decline. The ultimate metric is determined as the maximum IoU achieved.

The reliance on groundtruth to guide the methodology can introduce biases stemming from the performance of the deep network, which is not ideal. Although Zhang et al. [15] did not address this issue, our study explores the instance segmentation performance by examining various ranges of model confidence. We establish a prediction threshold by selecting a value between two specified bounds, and we apply the wrapping game to all instances where the model confidence falls within this threshold. This approach enables us to gain a more comprehensive understanding of the explainer's discriminatory power while also considering the impact of the deep network detector model's performance on the explanations generated by our method.

The trends in Figures 9b and 10b show the wrapping game analysis when considering different levels of model confidence (*prediction_thr*). These settings reveal a distinct relationship between the performance of the model and the Intersection over Union (IoU) score. Both explainers demonstrate enhanced scores in contexts where the model displays elevated confidence levels. This finding is consistent with the premise that a model's confidence is intrinsically linked to its ability to differentiate a particular instance from its surrounding environment.

Figure 10 presents the Wrapping Game analysis for the LRP-based framework. Although the explainer initially demonstrates strong performance, as shown in the Confidence-Agnostic Analysis (Figure 10a), its discriminative ability declines significantly on instances where the model's confidence is low (Figure 10b). This limitation is particularly critical, as explanations are most valuable in low-confidence scenarios, where understanding the model's decision-making process is essential.

The evaluation shown in Figure 9 highlights that the proposed DetDSHAP explainer achieves a reasonable IoU score on the dataset, albeit not as high as that achieved by YOLO LRP. However, unlike YOLO LRP, the DetDSHAP framework maintains its effectiveness even in instances where the model's confidence is low. This resilience makes DetDSHAP better suited for addressing the crucial low-confidence scenarios, where reliable explanations are indispensable.

4.2 | Pruning Performance

In this study, we evaluate the efficacy of our pruning framework and conduct a comparative analysis of its performance using DetDSHAP as a criterion in contrast to LRP. Our framework is applied to prune a variety of deep detector networks, and we illustrate the relationship between performance quality and structural efficiency through plotted data. These visual representations are displayed in Figures 11–16. Each figure comprises two distinct plots: the left plot depicts the network's performance relative to the reduction in GFLOPS, while the right plot illustrates the network's performance in relation to the percentage of parameters pruned. In the context of pruning deep network classifiers, the authors of ref. [25] report pruning 5% of their network's parameters and subsequently fine-tuning for 10 epochs at each pruning stage. In our approach, we select varying

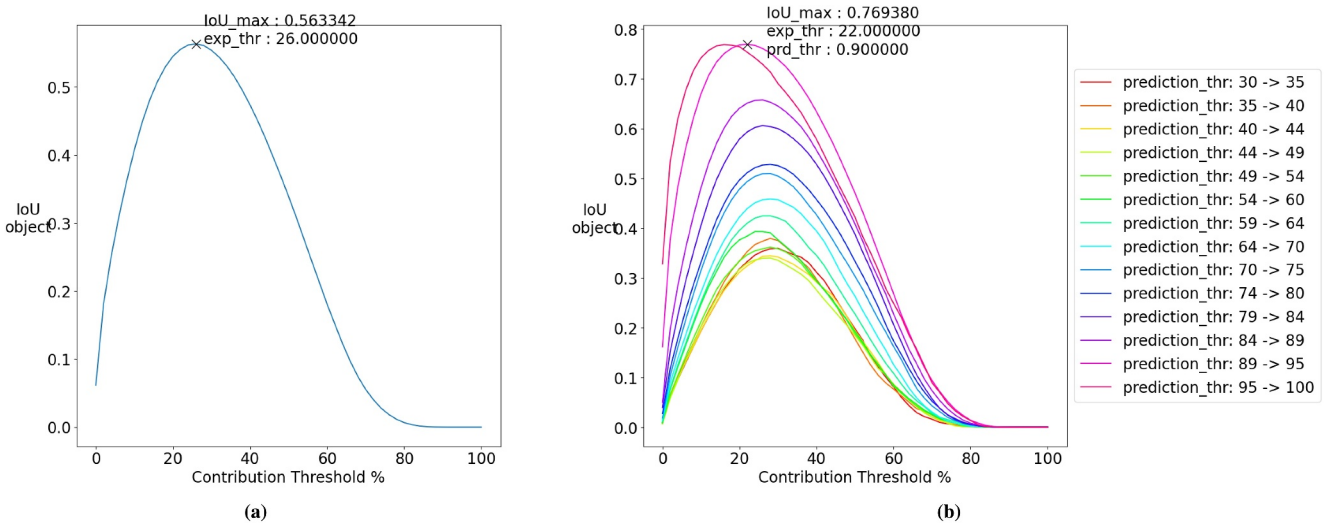


FIGURE 9 | The above plots illustrate the wrapping game analysis of the proposed DetDSHAP framework with YOLOV5. (a) Analysis across all instances in the set. (b) Analysis considering the model's confidence in a given instance.

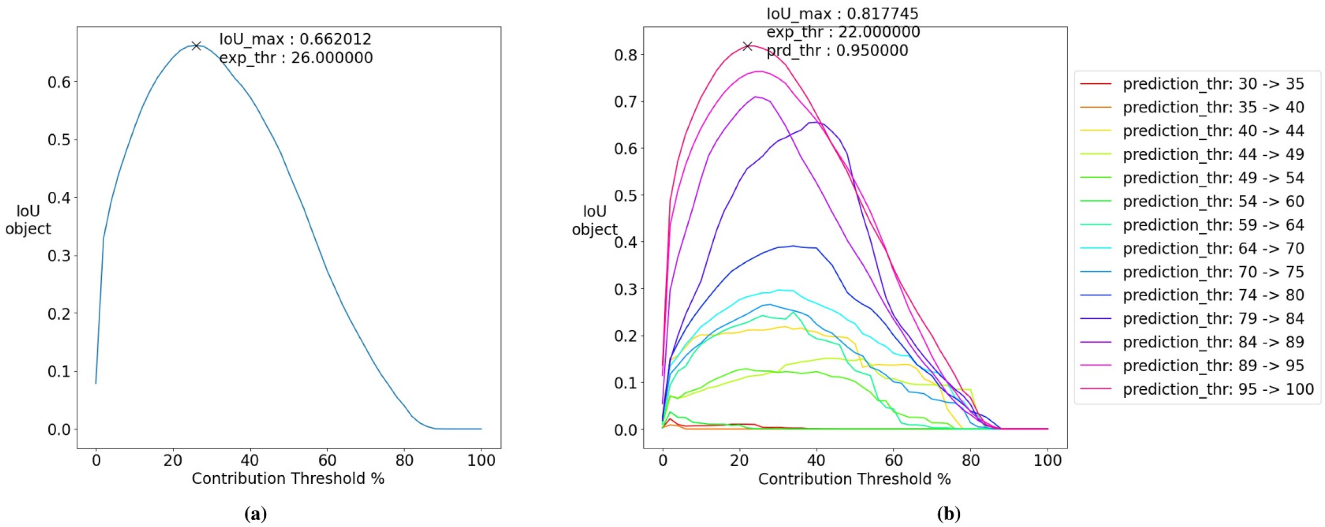


FIGURE 10 | The above plots illustrate the wrapping game analysis of the proposed YOLO LRP framework with YOLOV5. (a) Analysis across all instances in the set. (b) Analysis considering the model's confidence in a given instance.

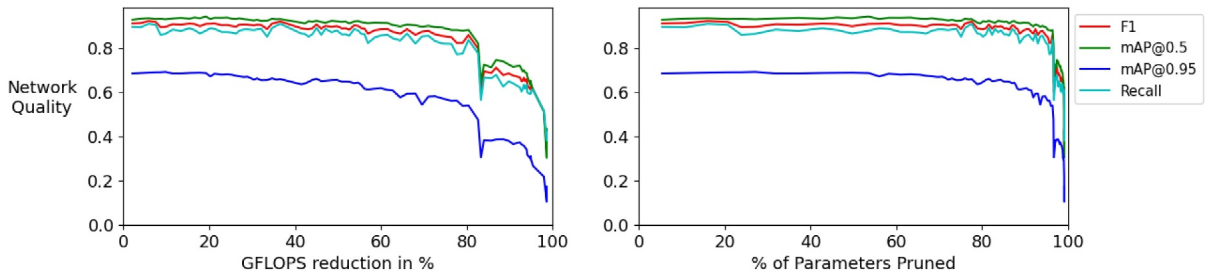


FIGURE 11 | This figure displays the pruning trend when using DetDSHAP as a criterion to prune YOLOv5s trained on our self-driving car dataset.

parameters tailored to each dataset to optimise performance for each specific scenario.

In the initial scenario, we implement pruning on the YOLOv5s model, which has been trained using data collected from our self-driving car platform. During each pruning iteration, we select a batch of 10 samples to establish the filter rankings and subsequently prune 2.5% of the total filters, followed by 10 epochs of fine-tuning after each pruning step. This process is executed utilising our DetDSHAP method to derive the filter

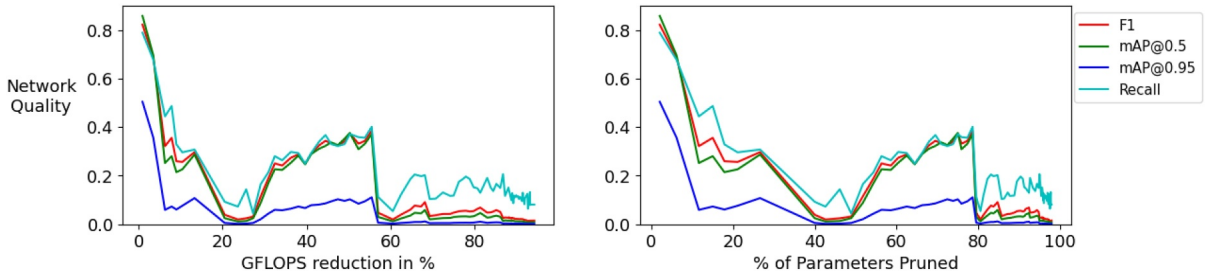


FIGURE 12 | This figure displays the pruning trend when using LRP as a criterion to prune YOLOv5s trained on our self-driving car dataset.

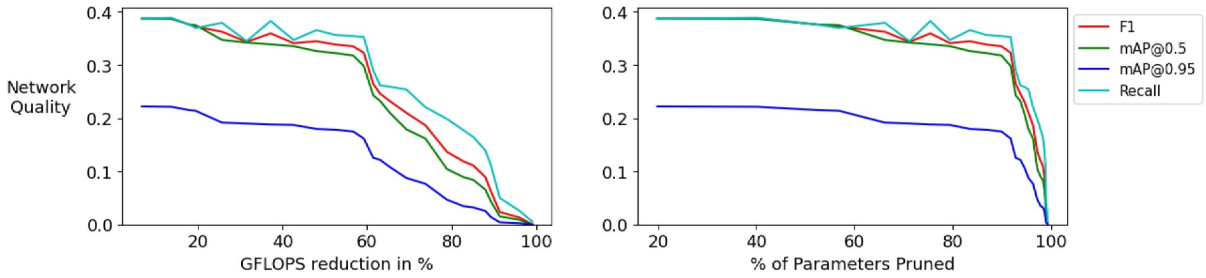


FIGURE 13 | This figure displays the pruning trend when using DetDSHAP as a criterion to prune YOLOv5l trained on the Visdrone dataset.

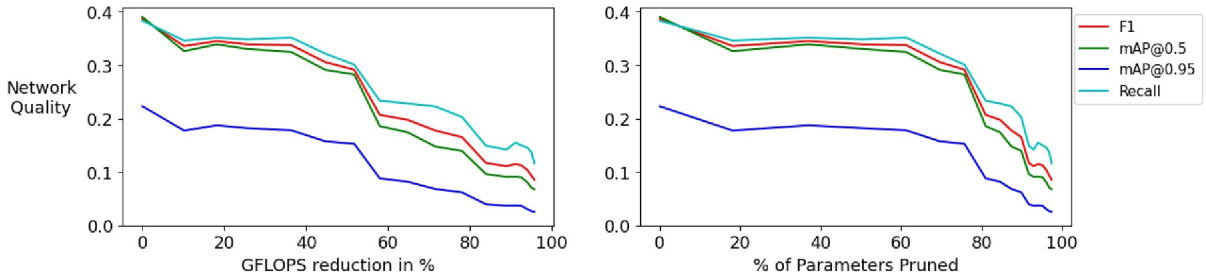


FIGURE 14 | This figure displays the pruning trend when using LRP as a criterion to prune YOLOv5l trained on the Visdrone dataset.

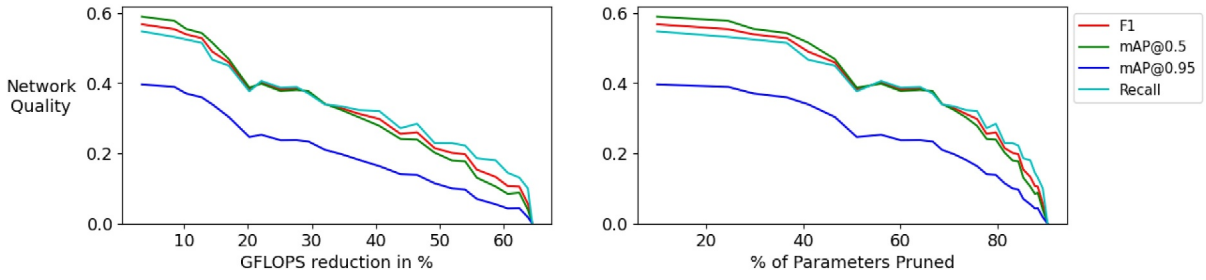


FIGURE 15 | This figure displays the pruning trend when using DetDSHAP as a criterion to prune YOLOv5s trained the COCO2017 dataset.

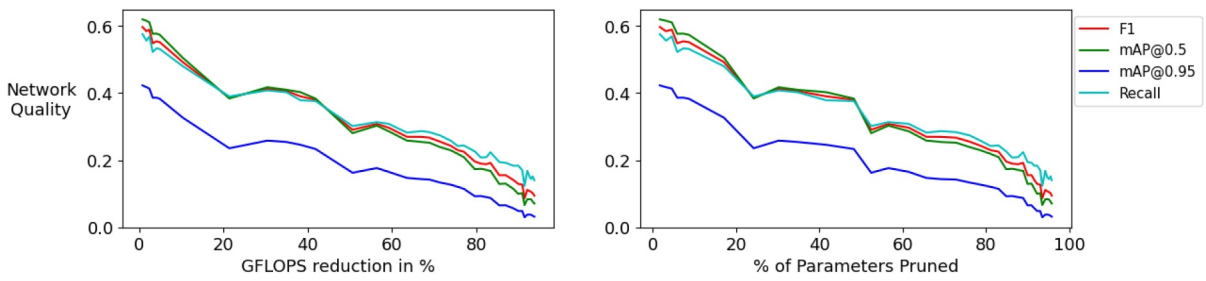


FIGURE 16 | This figure displays the pruning trend when using LRP as a criterion to prune YOLOv5s trained the COCO2017 dataset.

rankings. For comparative purposes, we also apply the same methodology using the LRP explainer as referenced in ref. [16]. The results of this analysis are illustrated in Figures 11 and 12, respectively.

Figure 11 indicates that DetDSHAP can discard almost all the redundant capacity in a single-class detector without harming accuracy. Pruning 80% of the parameters removes roughly 40% of the FLOPs yet the mAP curve remains flat. Indeed, performance only begins to fall once 95% of the parameters have been removed - this equates to a reduction of around 80% in FLOPs. The gentle slope confirms that the DetDSHAP score really does reflect filter utility for this task.

In contrast, the LRP baseline behaves very differently (Figure 12. Employing this criterion leads to an immediate decline in performance after the first 10% of filters. There are instances where performance improves—this is attibuted to subnetworks with favourable initialisations temporarily enhancing performance. Ultimately throughout the pruning process the performance fails to recover above 50% of the initial performance level. Notably, when 80% of the parameters are pruned, the model's performance is entirely compromised.

Figure 13 shows the effect of DetDSHAP - guided pruning on the large YOLOv5l model trained for the VisDrone 10-class task. Because this backbone is much deeper than YOLOv5s and each image contains many objects of widely varying scale, we prune 10% of the filters per iteration while keeping the batch size fixed. Despite the higher pruning rate, DetDSHAP can remove more than 60% of the parameters—a 22% drop in FLOPs—before any noticeable mAP loss occurs, and mAP stays within 3 pp of the baseline until 93% of filters are gone (≈60% FLOP reduction). This confirms that the relevance scores retain their discriminatory power even in a challenging multi-class setting.

Figure 14 contrasts the same experiment using LRP relevance as the pruning metric. LRP performs slightly better here than on the single-class scenario (Figure 12), possibly because its relevance is biased toward class-score channels; nevertheless, the initial accuracy dip and the steeper decline beyond 30% pruning leave it consistently below the DetDSHAP curve across the useful compression range. DetDSHAP therefore remains the preferred criterion for large, multi-class detector backbones.

Our final experiment applies DetDSHAP-guided pruning to YOLOv5m trained on COCO 2017. Because the dataset covers 80 object classes—with each image averaging 7.2 instances—we increase the sampling pool to 100 images per pruning step to give every class a fair chance of influencing the filter ranks. The pruning rate is set to 5% per iteration and followed by 10 epochs of fine-tuning, as before.

Figure 15 shows that DetDSHAP holds the detector's mAP essentially constant until about 25% of the parameters are removed, yielding a 10% FLOP reduction. Accuracy then degrades gracefully, but once pruning exceeds 60% of the parameters the mAP falls by roughly 40 pp. This steep tail suggests that extreme compression is limited by class imbalance and by the very small objects prevalent in COCO. Accuracy then degrades gradually, but once pruning exceeds 60% of the parameters the mAP falls by roughly 40 pp. Figure 16 plots the same experiment with LRP relevance: that curve drops sooner and more steeply, confirming the pattern seen on VisDrone. Future work will explore class-balanced sampling and focal regularisation during the rank-accumulation step to stabilise importance estimates in such diverse datasets.

The trend observed in our final scenario is illustrated in the plots presented in Figure 15. In this scenario, it is evident that the decline in performance is more pronounced. Specifically, at the point where 50% of the parameters have been pruned, each equality metric experiences a reduction of approximately 40%. However, it remains feasible to prune roughly 25% of the parameters, which corresponds to a 10% enhancement in the GFLOPs, while exerting minimal impact on the quality performance of the deep network detector.

Using LRP as a criterion may yield more favourable outcomes in multi-class scenarios. This is particularly evident when contrasting the trends observed in our single-class dataset (as illustrated in Figure 12) with those from the Visdrone 10-class dataset. Such observations suggest that the underlying explainer exhibits a bias towards the class score. Although LRP demonstrates enhanced performance, there is a significant initial decline in the network's quality. Consequently, the quality throughout the subsequent pruning process remains inferior compared to when DetDSHAP is employed as the criterion. This pattern is similarly reflected in Figure 16. Furthermore, this latter scenario indicates that it is possible to prune beyond DetDSHAP when using LRP as the criterion; however, this extension is limited to approximately 10% beyond DetDSHAP, and at this juncture, the original performance of the network is severely compromised, rendering it likely unusable.

4.3 | Post-Pruning Network Performance Quality

In this sub-section, we conduct a thorough examination of the impact of our pruning framework on the performance quality of the YOLOv5 detection models used in this research. We have selected three deep detection models for further development during the pruning process. Each pruned model undergoes an additional fine-tuning phase for 200 epochs. The quality of our final network, along with their mean average precision (mAP), is presented in Table 3, where we also include efficiency scores. These efficiency scores encompass not only the number of parameters and floating-point operations per second FLOPs but also the inference times measured on an NVIDIA GeForce RTX 2060S. To enhance our analysis, we present the Precision-Recall curves (PR curves) for the models trained on the three datasets employed in this study. This graphical representation of precision against recall serves as a critical evaluation of the performance quality of the object detection models. A deep detection network is deemed effective if it maintains high precision while recall increases.

In the first column of Table 3, we present the final outcomes of our pruned YOLOv5s model, specifically tailored for our self-driving car dataset. The results indicate that following additional fine-tuning, there is only a negligible decrease in the

mAP@0.5–0.95 score, along with a slight enhancement in the mAP@0.5. This is noteworthy given that there has been a reduction of more than 80% in the number of parameters and a 40% decrease in GFLOPS. Furthermore, Figure 17a illustrates the precision-recall curves for the models trained on our self-driving car dataset. The comparison reveals minimal variation in the PR curve between the unpruned and pruned models, despite the significant reductions in parameters and FLOPs detailed in Section 4.2. As previously mentioned, our self-driving car dataset is relatively straightforward, contrasting with more complex scenarios, which is reflected in our findings.

The findings presented in the second column of Table 3 indicate that a comparable outcome can be achieved for a network trained on the more complex Visdrone dataset. Notably, we successfully decreased the number of parameters in our YOLOv5l network by nearly 50% and reduced the GFLOPs by 16%, all while maintaining performance levels that are not significantly compromised. Additionally, Figure 17b illustrates the precision-recall curve for our models evaluated on the Visdrone dataset.

In the context of Visdrone, we further illustrate the confusion matrices obtained from both the pruned and unpruned models in Figure 18. The predominant class detected in this dataset for both models is 'car', which exhibits the least decline in the True Positive (TP) rate and is the most frequently occurring class within the dataset. However, a significant drawback of the pruned model is the increased proportion of false negatives, as depicted in Figure 18. This suggests a decline in recall performance; nevertheless, an evaluation of the PR curve in Figure 17b indicates that this decline is not considerable.

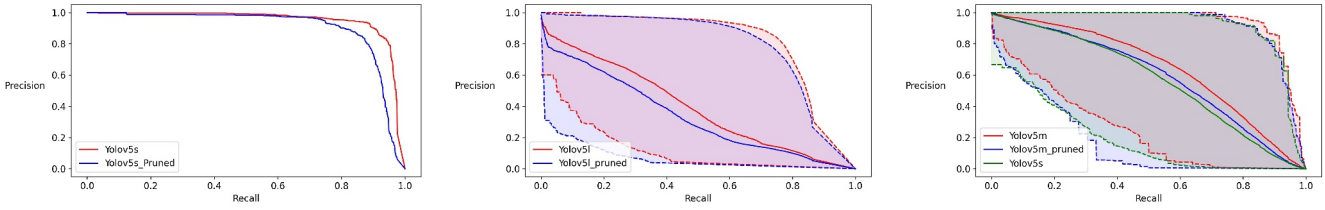
Table 3 presents the outcomes of pruning the YOLOv5m model, which was trained on the COCO2017 dataset. Additionally, Figure 17c illustrates the precision-recall (PR) curves for both the pruned YOLOv5m model and the unpruned version. In this analysis, we also include the PR curve for a YOLOv5s model trained on the same dataset. The data indicates that, in contrast to the other scenarios discussed in this section, there is a notable decline in the network's performance, with a reduction exceeding 10% in the mean average precision (mAP) at the 0.5–0.95 threshold. Nevertheless, the PR curves depicted in Figure 17c reveal that the pruned YOLOv5m model outperforms the YOLOv5s model on average across all classes. However, it is important to note that the performance diminishes in the class with the lowest performance. This trend is also evident in Figure 17b and is likely attributable to class imbalance, which affects the filter rankings. These rankings are crucial as they dictate which components of the network should be pruned, and in our proposed methodology, they are derived from samples taken from the dataset. Future research could explore more advanced sampling techniques to mitigate the bias in filter rankings towards any specific class.

4.4 | Pruning Verses Design

A frequently neglected aspect in the existing literature is the comparative effectiveness of pruning versus initiating with a more straightforward architecture. In this section, we explore this issue by contrasting our pruned YOLOv5m, which has been trained on the COCO2007 dataset, with YOLOv5s which has undergone training on the same dataset. For this analysis, we

TABLE 3 | The performance comparison between the unpruned and pruned YOLOv5 models. All measurements were taken on NVIDIA GeForce RTX 2060 super.

	Self-driving car data			Visdrone			COCO2017		
	YOLOv5s	YOLOv5s pruned	% change	YOLOv5l	YOLOv5l pruned	% change	YOLOv5m	YOLOv5m pruned	% change
mAP_{50-95}	70.7	68.8	−2.69	23.1	22.1	−3.90	44.1	39.1	−11.90
mAP_{50}	94.7	94.8	+0.11	39.6	38.6	−2.52	63.5	58.6	−7.71
Params(M)	7.01	1.30	−81.46	46.3	23.9	−48.38	21.2	6.5	−69.34
$GFLOPs_{512}$	10.08	5.91	−41.36	69.4	58.2	−16.14	48.9	33	−32.52
Infer Time(ms)	1.62	1.22	−25.0%	6.40	5.67	−11.4%	5.38	4.38	−18.59%



(a) Our Dataset **(b)** Visdrone **(c)** COCO2017

FIGURE 17 | The plots in this figure depict the precision-recall curves before and after pruning for the YOLO networks developed for each dataset used in this study. For the plots in (b) and (c), the solid lines denote the average trend for all classes in their respective dataset, and the dotted lines represent the best and worst cases across all classes.

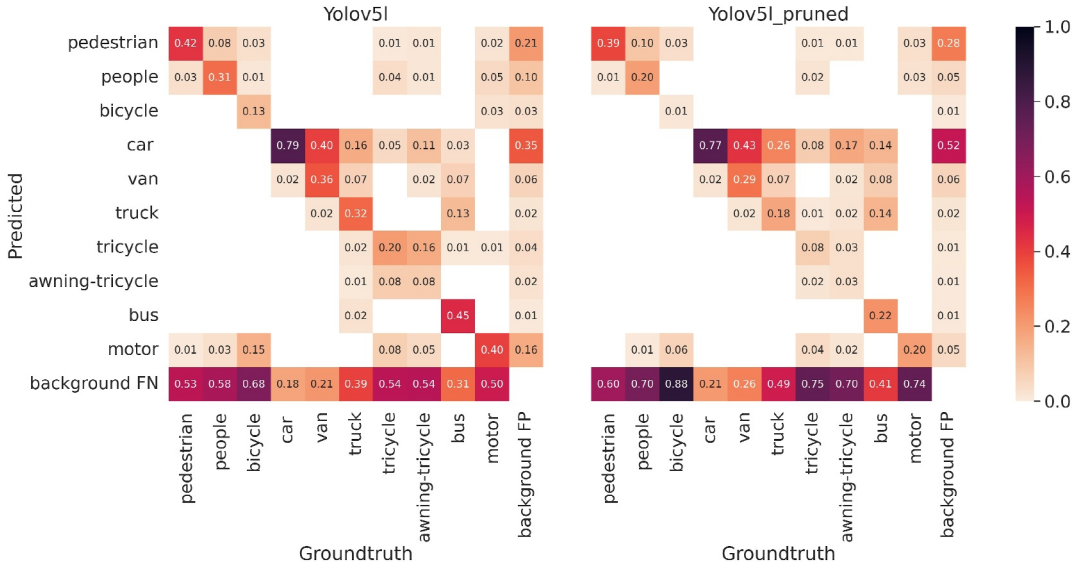


FIGURE 18 | Confusion matrix of the unpruned YOLOv5l network trained on the Visdrone dataset.

restricted our focus to publicly available pretrained models from ref. [8] to eliminate any biases that might arise from our custom implementation.

The bar graph presented in Figure 19 illustrates the comparative performance of our pruned YOLOv5m against its unpruned counterpart and YOLOv5s. The performance metrics for the unpruned models are sourced from ref. [8]. Initially, YOLOv5m comprised 21.2 million parameters, which we subsequently reduced to 6.5 million parameters through pruning. This figure is significantly lower than the parameter count of YOLOv5s. Furthermore, our pruned YOLOv5m demonstrates superior performance in both mAP metrics.

YOLOv5s demonstrates superior performance in GFLOPs when compared to our pruned YOLOv5m. This discrepancy can be attributed to our pruning criterion, which predominantly emphasised the upper layers nearer to the model's output. However, if the objective of pruning is to minimise the FLOPs required during inference, it would be more advantageous to concentrate on pruning the lower layers that are closer to the model's input. Future research will aim to modify the pruning framework to prioritise these layers, thereby enabling a more aggressive reduction of FLOPs.

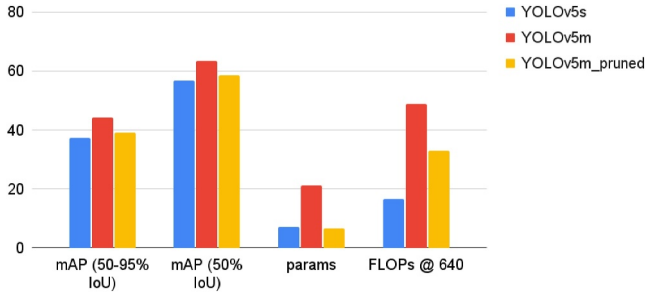


FIGURE 19 | This plot shows the performance quality and structure efficacy of our pruned YOLOv5m model compared to two other models released by [8]. Training and evaluation are conducted on the COCO2017 dataset.

5 | Conclusion

This research illustrated the intersection of explainability and pruning within the YOLOv5 framework. The DetDSHAP methodology was proposed as a means to investigate a neural network, thereby enhancing comprehension of its decision-making processes and enabling designers to identify the constraints of their deep learning models. This concept was further developed by showcasing DetDSHAP as a viable criterion for pruning, resulting in the removal of a substantial portion of the parameters from deep networks with negligible impact on performance—nearly 50% on the Visdrone dataset and over 80% on the single-class self-driving car dataset. Additionally, the pruning framework presented in this chapter demonstrated that a large network could achieve greater memory efficiency compared to a similarly performing smaller network. Ultimately, through the analysis of the wrapping game and the demonstrated efficacy as a pruning criterion, it can be confidently asserted that DetDSHAP accurately allocates causal information.

Improvements can be made to the existing pruning framework to enhance the quality of models following the pruning process. In a manner akin to the findings of Yeom et al. [25], the criterion proposed here emphasises layers that are situated closer to the network's output, which consequently diminishes its effectiveness in minimising the number of floating-point operations per second (FLOPS). Furthermore, the authors suggest that modifying the selection process for samples used in determining filter ranks could lead to a reduced decline in performance after pruning. This could involve strategies such as ensuring a balance in class labels among the samples, as well as considering the positioning and dimensions of their bounding boxes. A deeper exploration of this topic will form the foundation for future research, as examining the model's responses to various data groupings may yield insights into its operational behaviour and facilitate the development of comprehensive explanations,

similar to those identified by Lapuschkin et al. [37]. Such explanations offer a more integrated perspective on the model's decision-making processes.

Author Contributions

Maxwell Hogan: conceptualization, data curation, formal analysis, methodology, project administration, software, validation, visualization, writing – original draft, writing – review and editing. **Nabil Aouf:** conceptualization, formal analysis, methodology, project administration, supervision, writing – review and editing.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

1. M. Abdulsalam and N. Aouf, "Deep Weed Detector/Classifier Network for Precision Agriculture," in *2020 28th Mediterranean Conference on Control and Automation (MED)*, (2020), 1087–1092.

2. P. Pyrrö, H. Naseri, and A. Jung, "Rethinking Drone-Based Search and Rescue With Aerial Person Detection," *ArXiv* (2021): 09406: abs/2111.

3. O. Kechagias-Stamatis, N. Aouf, and D. Nam, "3D Automatic Target Recognition for UAV Platforms," in *2017 Sensor Signal Processing for Defence Conference (SSPD)*, (2017), 1–5.

4. M. Hogan, N. Aouf, P. Spencer, and J. Almond, "Explainable Object Detection for Uncrewed Aerial Vehicles Using KernelSHAP," in *2022 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, (2022), 136–141.

5. F. Wang and C. Rudin, "Causal Falling Rule Lists," in *Proceedings of Workshop on Fairness, Accountability, and Transparency in Machine Learning* (Halifax, Nova Scotia, 2017).

6. P. W. Koh and P. Liang, "Understanding Black-Box Predictions via Influence Functions," in *ICML'17. Proceedings of the 34th International Conference on Machine Learning - Volume 70* (JMLR.org, 2017), 1885–1894.

7. R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization," *International Journal of Computer Vision* 128, no. 2 (2019): 336–359, https://doi.org/10.1007/s11263-019-01228-7.

8. G. Jocher, A. Stoken, and J. Borovec, *Ultralytics/YOLOv5* (GitHub repository, 2021), https://github.com/ultralytics/yolov5.

9. H. Chen, S. M. Lundberg, and S. I. Lee, "Explaining a Series of Models by Propagating Shapley Values," *Nature Communications* 13, no. 1 (2022): 4512, https://doi.org/10.1038/s41467-022-31384-3.

10. L. He, N. Aouf, and B. Song, "Explainable Deep Reinforcement Learning for UAV Autonomous Path Planning," *Aerospace Science and Technology* 118 (2021): 107052, https://doi.org/10.1016/j.ast.2021.107052.

11. v. dB. H. Velden, M. A. A. Ragusi, M. H. A. Janse, C. E. Loo, and K. G. A. Gilhuijs, "Interpretable Deep Learning Regression for Breast Density Estimation on MRI," in *Medical Imaging 2020: Computer-Aided Diagnosis* (SPIE, 2020).

12. S. M. Lundberg and S. I. Lee, "A Unified Approach to Interpreting Model Predictions," in *NIPS'17. Proceedings of the 31st International Conference on Neural Information Processing Systems* (Curran Associates Inc., 2017), 4768–4777.

13. V. Petsiuk, A. Das, and K. Saenko, "RISE: Randomized Input Sampling for Explanation of Black-Box Models," *ArXiv* (2018): abs/1806.07421.

14. V. Petsiuk, R. Jain, V. Manjunatha, et al., "Black-Box Explanation of Object Detectors via Saliency Maps," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2020), 11438–11447.

15. J. Zhang, S. A. Bargal, Z. Lin, J. Brandt, X. Shen, and S. Sclaroff, "Top-Down Neural Attention by Excitation Backprop," *International Journal of Computer Vision* 126, no. 10 (2018): 1084–1102, https://doi.org/10.1007/s11263-017-1059-x.

16. A. Karasmanoglou, M. Antonakakis, and M. Zervakis, "Heatmap-Based Explanation of YOLOv5 Object Detection With Layer-Wise Relevance Propagation," in *2022 IEEE International Conference on Imaging Systems and Techniques (IST)*, (2022), 1–6.

17. M. Sabih, F. Hannig, and J. Teich, "Utilizing Explainable AI for Quantization and Pruning of Deep Neural Networks," *ArXiv* (2020): 09072: abs/2008.

18. T. Y. Lin, M. Maire, S. J. Belongie, et al., "Microsoft COCO: Common Objects in Context," in *European Conference on Computer Vision*, (2014).

19. K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps," *CoRR* (2013): abs/1312.6034.

20. M. Hogan and P. N. Aouf, "Towards Real Time Interpretable Object Detection for UAV Platform by Saliency Maps," in *2021 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, (2021), 1178–1183.

21. G. Montavon, A. Binder, S. Lapuschkin, W. Samek, and K. R. Müller, *Layer-Wise Relevance Propagation: An Overview* (Springer International Publishing, 2019), 193–209.

22. H. Tsunakawa, Y. Kameya, H. Lee, Y. Shinya, and N. Mitsumoto, "Contrastive Relevance Propagation for Interpreting Predictions by a Single-Shot Object Detector," in *2019 International Joint Conference on Neural Networks (IJCNN)*, (2019), 1–9.

23. A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *ArXiv* (2020): 10934: abs/2004.

24. A. Shrikumar, P. Greenside, and A. Kundaje, "Learning Important Features through Propagating Activation Differences," in *International Conference on Machine Learning*, (2017).

25. S. K. Yeom, P. Seegerer, S. Lapuschkin, et al., "Pruning by Explaining: A Novel Criterion for Deep Neural Network Pruning," *Pattern Recognition* 115 (2021): 107899, https://doi.org/10.1016/j.patcog.2021.107899.

26. D. Lundstrom, A. Huyen, A. Mevada, K. Yun, and T. Lu, "Explainability Tools Enabling Deep Learning in Future In-Situ Real-Time Planetary Explorations," in *2022 IEEE Aerospace Conference (AERO)*, (2022), 1–8.

27. M. M. Pasandi, M. Hajabdollahi, N. Karimi, and S. Samavi, "Modeling of Pruning Techniques for Deep Neural Networks Simplification," *arXiv* (2020): 04062: abs/2001.

28. M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic Attribution for Deep Networks," in *International Conference on Machine Learning*, (2017).

29. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification With Deep Convolutional Neural Networks," *Communications of the ACM* 60, no. 6 (2017): 84–90, https://doi.org/10.1145/3065386.

30. K. Wagstaff, S. Lu, E. Dunkel, et al., "Mars Image Content Classification: Three Years of NASA Deployment and Recent Advances,"

Proceedings of the AAAI Conference on Artificial Intelligence 35, no. 17 (2021): 15204–15213: IAAI Technical Track on Highly Innovative Applications of AI, https://doi.org/10.1609/aaai.v35i17.17784.

31. D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Guttag, "What Is the State of Neural Network Pruning?," *Proceedings of machine learning and systems* 2 (2020): 129–146.

32. N. Lee, T. Ajanthan, and P. Torr, "SNIP: Single-Shot Network Pruning Based on Connection Sensitivity," in *International Conference on Learning Representations*, (2019).

33. S. Bach, A. Binder, G. Montavon, F. Klauschen, K. R. Müller, and W. Samek, "On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation," *PLoS One* 10, no. 7 (2015): e0130140, https://doi.org/10.1371/journal.pone.0130140.

34. M. Hogan and P. N. Aouf, "Explainable Dataset for Ground Based Drones in Urban Environments," in *2024 IEEE International Conference on Robotics and Biomimetics (ROBIO)* (Accepted for publication, 2024).

35. P. Zhu, L. Wen, D. Du, et al., "VisDrone-DET2018: The Vision Meets Drone Object Detection in Image Challenge Results," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, (2018).

36. J. Sklansky, "Finding the Convex Hull of a Simple Polygon," *Pattern Recognition Letters* 1, no. 2 (1982): 79–83, https://doi.org/10.1016/0167-8655(82)90016-2.

37. S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, and K. Müller, "Unmasking Clever Hans Predictors and Assessing What Machines Really Learn," *CoRR* (2019): 10178: abs/1902.