Neurocomputing 611 (2025) 128656

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Saliency and location aware pruning of deep visual detectors for autonomous driving

Jung Im Choi [a], Qing Tian [a,b,*]

[a] Department of Computer Science, Bowling Green State University, Bowling Green, OH, 43403, USA
[b] Department of Computer Science, University of Alabama at Birmingham, Birmingham, AL, 35294, USA

ARTICLE INFO

Communicated by F. Khelifi

Keywords:
Deep detector pruning
Efficient self-driving perception
Saliency-location-aware channel importance

ABSTRACT

Despite the remarkable achievements of deep neural networks, their high computational complexity limits their wide use in many real-world embedded applications, such as autonomous driving perception. While current neural network pruning approaches can reduce model complexity to various extents, they often adopt local or ad hoc importance measures that are not directly related to the final task. More importantly, most of them focus on classification tasks and do not take location information into consideration during pruning. To address these issues, we present a novel channel importance measure that incorporates detection-related saliency and location awareness, specifically designed for the pruning of self-driving visual detectors. Our comprehensive experiments on the KITTI and COCO_traffic datasets demonstrate that our pruning method achieves significant reductions in model size and computational operations with little performance degradation. Notably, it outperforms other state-of-the-art methods across various pruning rates and base detectors. Our pruned YOLOX-S model with 40.2% fewer parameters even improves the original model's mAP by 1.8% on KITTI. Moreover, we experimentally highlight the potential of our pruning approach in effectively detecting small-scale objects.

1. Introduction

Deep neural networks (DNNs) have advanced the state of the art across various computer vision tasks. However, their huge size and computational costs prevent their wide deployment on resource-constrained platforms. Especially in autonomous driving, long processing time can be disastrous. Given that autonomous driving algorithms usually run on resource-constrained hardware (e.g., embedded CPUs/GPUs), compressing deep models becomes crucial to achieving real-time performance. Neural network pruning has become one the most popular and widely used compression techniques [1–3]. Unlike other compression techniques, such as knowledge distillation [4,5] and quantization [6,7]), network pruning directly removes unnecessary/redundant components from a neural network [1,2,8–10]. Especially, structured pruning which aims to remove entire filters or neurons has gained great attention due to its direct speedup and compression using general-purpose hardware or Basic Linear Algebra Subprograms (BLAS) libraries [2,10–14].

The main difference among various pruning methods lies in their importance measure. For instance, many pruning works [2,11,13–15] based on straightforward statistics of filter weights, such as magnitude or variance, tend to rely on locally computed importance measures. They often overlook the interconnections among filters and/or have little relevance to the final task utility. Furthermore, it is worth noting that most pruning approaches are designed for classification tasks [11,12,14,16,17] and do not pay special attention to the location information, which is crucial to visual detection. In this paper, we propose a novel saliency-and-location-aware channel pruning technique that is specially designed for self-driving visual detection. The key contributions of this paper are summarized as follows:

- Unlike existing pruning approaches, our channel importance measure directly reflects a channel's contribution to the final detection utility. We leverage gradients derived from detection losses with respect to channel features to help guide the pruning process of deep visual detection models.
- We take critical location information into consideration during deep detector pruning. We find that both actual object location information and contextual information surrounding the objects are beneficial.
- In our experiments on KITTI and COCO_traffic, we demonstrate that our proposed approach can beat state-of-the-art (SOTA) pruning approaches. For example, on KITTI, the proposed approach

* Corresponding author at: Department of Computer Science, University of Alabama at Birmingham, Birmingham, AL, 35294, USA.
E-mail address: qtian@uab.edu (Q. Tian).

https://doi.org/10.1016/j.neucom.2024.128656
Received 25 October 2023; Received in revised form 11 June 2024; Accepted 20 September 2024
Available online 26 September 2024
0925-2312/© 2024 Elsevier B.V. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

attains substantial parameter reductions of 36.9% and 59.7% or FLOP reductions of 43.1% and 33.6%, with only minor mAP drops of 0.2% and 0.1% for YOLOF and ATSS, respectively. Notably, our pruned model can even beat the original unpruned model. For example, in the YOLOX-S on KITTI case, we achieve a 1.8% improvement in mAP while reducing parameters and FLOPs by 40.2% and 29.2%, respectively.
- Our pruning method performs particularly well in detecting small-scale objects, which is an essential aspect for practical applications like autonomous driving.

This is the journal version of our previous work [18], with significant improvements: (1) We demonstrate the efficacy of our approach in comparison to a wider range of cutting-edge detection models [19–22]. (2) To show the advantages of our method, we compare our method with HRank [17] and HALP [23], in addition to L1 Pruning [2], Net Slimming [24], and CHIP [16]. (3) We conduct a comprehensive qualitative analysis employing various pruning methods for comparison, which offers an in-depth visual understanding of our proposed approach's advantages. (4) Through extensive experiments, we investigate the behavior of our pruned models across various pruning rates and choose a decay function that better captures contextual information for effective pruning. (5) We also explore the real-world consequences of parameter and FLOP reduction through pruning and how our pruning affects inference latency.

2. Related work

2.1. Deep visual detectors

Visual detection is one critical component in autonomous driving systems, which aims to precisely locate and identify road objects such as cars, pedestrians, and traffic signs, making the entire driving process safer. Compared to traditional visual detectors [25,26], deep visual detectors usually provide better performance in finding and localizing these objects [21,27–30]. Typically, deep visual detection algorithms can be classified into two main types: one-stage and two-stage. Two-stage detectors like the R-CNN family [27,31,32] have a region of interest (ROI) proposal step followed by classification and bounding box regression of each candidate ROI. These models can offer comparable or better accuracy but are slower than one-stage detectors. On the other hand, one-stage detectors like the YOLO family [21, 22,28,33] skip the region proposal stage and run classification and localization directly and simultaneously, making them more efficient than two-stage detectors. In this work, targeting the time-sensitive task of autonomous driving perception, we explore various one-stage visual detectors (e.g., YOLOX-S [21], YOLOF [22], GFL [20], ATSS [19]) as our baseline models and aim to further improve the model deployment efficiency through pruning.

2.2. Pruning of neural networks

As neural networks become more complex, their memory and computational requirements grow substantially. Consequently, neural network pruning has become a prominent strategy to improve network efficiency. Pruning methods remove redundant or unnecessary components within neural networks and can be roughly classified into unstructured [1,8,9,15,34] and structured pruning [2,10–14,17].

Unstructured pruning. Early works, like Optimal Brain Damage [8] and Optimal Brain Surgeon [9], focused on unstructured pruning by removing individual weights. While these methods demonstrated the feasibility of removing unnecessary weights with minimal loss in accuracy, they face challenges when being adapted to deep networks due to computational costs or strict assumptions. Later unstructured pruning approaches include [1,15,34–36]. Han et al. [1] presented a magnitude-based method to eliminate unimportant weights with absolute values smaller than a predefined threshold. Guo et al. [15] introduced dynamic network surgery, allowing pruned weights to be restored if deemed crucial, to compensate for unexpected loss. Recently, Park et al. [36] introduced lookahead pruning as an extension of magnitude-based pruning. Nevertheless, the practical implementation of these unstructured approaches presents challenges within the existing hardware and software environments [37,38].

Structured pruning. In contrast to unstructured pruning, structured pruning methods [2,10–14,16,17,39] directly removes entire channels or filters, enabling actual speedup and compression using off-the-shelf hardware or software. The key difference among the existing structured pruning methods lies in how they assess the importance of a filter/channel. Li et al. [2] proposed a filter pruning method based on the L1-norm of the filter weights, considering filters with smaller L1-norm values as less important and pruning them. He et al. [14] presented a geometric median-based pruning approach, where the filters closest to this median were considered redundant and subsequently removed. While these filter-weight-based methods are simple and effective, they do not consider the distribution of input data, potentially leading to suboptimal performance. On the other hand, activation/feature-based methods consider both input data and filter parameters for filter pruning, allowing them to capture useful information from the actual data distribution and filters. Our proposed technique aligns with this line of research. Hu et al. [10] proposed to explore sparsity in activation maps by evaluating the importance of each filter based on the Average Percentage of Zero (APoZ). Liu et al. [24] proposed Network Slimming, which applies L1 regularization on the scaling factors in Batch Normalization layers to identify and prune less important channels. This method aims to induce sparsity in the scaling factors, which are then used to determine which channels to prune. In He et al. [40] and Luo et al. [41], they viewed channel selection as an optimization problem. Both methods [40,41] tried to minimize the reconstruction error of feature maps, where LASSO regression and greedy strategy were used to select the pruned channels, respectively. Liu and Wu [39] presented a channel pruning technique that relies on the mean gradient of feature maps in each layer. Lin et al. [17] presented HRank that prunes filters based on the rank of their feature maps, assuming lower-ranked feature maps contain less information. Sui et al. [16] proposed Channel Independence-Based Pruning (CHIP) to determine the importance of each feature map. Less independent channels are considered redundant and pruned. Shen et al. [23] introduced Hardware-Aware Latency Pruning (HALP), which aims to prune neural networks to meet specific hardware latency constraints while maximizing accuracy. However, these methods primarily target classification tasks and do not explicitly account for the spatial importance of features in detection tasks. Recently, Huang et al. [42] introduced CP3, a Channel Pruning Plug-in designed to enhance existing channel pruning methods on 3D Point-based networks. Guo et al. [43] introduced an automatic pruning approach based on the Information Bottleneck (IB) theory, addressing the channel pruning problem from an IB perspective.

Pruning for object detection. While most existing pruning approaches are tailored for deep classification models [2,16,17,24], there are only a limited number of studies addressing the more complex object detection tasks [44,45]. Targeting object detection, Xie et al. [44] presented a localization-aware channel pruning approach (LCP), which involves an auxiliary network designed for the object detection task. Li et al. [45] introduced a multi-task information fusion method for object detection pruning (MIFCP) where the multi-task information is extracted from an auxiliary network. Although both our approach and these methods focus on channel pruning for the object detection task, our approach differs in its comprehensive incorporation of visual saliency and location information, and the explicit inclusion of contextual information surrounding the objects. Unlike LCP [44] and MIFCP [45], which rely on auxiliary networks designed for object detection, our approach eliminates the need for such networks. These auxiliary networks increase

both the complexity of the pruning process and the overall training time. In contrast, our pruning importance measure is directly tied to the final detection objective, incorporating both saliency and location information. This measure can be computed with minimal overhead, streamlining the pruning process.

2.3. Visual saliency

Classic visual saliency methods provide insights into the decision-making of a neural network. Early saliency methods in the deep learning era include gradient-based approaches [46] and the deconvolutional network approach [47] to discover salient regions. Later methods like guided backpropagation [48], class activation mapping (CAM) [49], Grad-CAM [50], and Grad-CAM++ [51] were introduced to enhance saliency measures. All the above saliency measures are designed for classification tasks. Some works attempt to "predict" saliency with priors and assumptions. For example, Jian et al. [52] proposed integrating spatial position priors with background cues for visual saliency detection. The predicted saliency based on priors and assumptions may not align with the features a neural network deems important for decision-making. This paper introduces a saliency measure directly related to visual detection and incorporates it into channel importance calculation, along with location and context information, to guide the pruning of deep autonomous driving detectors.

3. Methodology

In this section, we present a saliency-and-location-aware channel pruning approach for deep visual detection models. Pruning deep visual detectors is more challenging than pruning deep classifiers, as visual detection involves both object classification and localization. Our goal is to minimize the complexity of deep visual detectors without degrading the performance (mAP).

3.1. Overview

Unlike the majority of pruning approaches that focus on classification, in this work, we attempt to prune visual detection models in a saliency-and-location-aware way, utilizing both classification and localization information. We particularly target autonomous driving applications where both accuracy and speed are critical. To quantify channels' detection utility, we consider the derivatives of the detection losses (including classification and localization regression) with respect to channel features. In addition, we integrate relevant object location information and contextual insights into our importance measure to make our pruning of object detectors location-aware. Unlike existing pruning approaches, our approach is holistic and captures each channel's contribution to the final detection utility. The overview of the proposed saliency and location aware pruning approach is shown in Fig. 1.

3.2. Saliency and location aware channel importance criterion

As discussed in Section 2.2, one major limitation with current pruning importance measures is their lack of direct alignment with task utility. We argue that the importance measure of individual units within a deep visual detector should consider their contribution to the final detection (including both classification and localization). In this paper, we propose a saliency and location aware detection utility measure and utilize it to guide our pruning.

To compute the channel importance score S^{kl} of the kth channel within layer l, we first define the detection-utility-weighted feature at the position (i, j) in the kth channel of layer l as:

$$w^{kl}_{(i,j)} = \Big(\frac{1}{WH}\sum_{i=1}^{W}\sum_{j=1}^{H}\frac{\partial \mathcal{L}_{det}}{\partial A^{kl}_{(i,j)}}\Big)A^{kl}_{(i,j)}, \tag{1}$$

where $\mathcal{L}_{det}$ is the total loss for the object detection task, $A^{kl}_{(i,j)}$ represents the feature point located at (i, j) within the kth feature map of layer l, W and H indicate the width and height of the feature map, respectively. The gradients of the detection utility with respect to channel features are averaged across the width and height dimensions, and then they are multiplied with the feature map to yield the detection-utility-weighted map denoted as $w^{kl}_{(i,j)}$. The process of pruning visual detectors should consider both classification and location aspects. The overall detection loss $\mathcal{L}_{det}$ is comprised of the classification loss $\mathcal{L}_{cls}$ and the bounding box regression loss $\mathcal{L}_{box}$:

$$\mathcal{L}_{det} = \sum_{m=1}^{M}\mathbb{I}^{obj}_{m}(\lambda_{cls}\mathcal{L}_{cls} + \lambda_{box}\mathcal{L}_{box}), \tag{2}$$

where $\mathbb{I}^{obj}_{m}$ indicates whether the mth bounding box contains an object (i.e., $\mathbb{I}^{obj}_{m} = 1$) or not (i.e., $\mathbb{I}^{obj}_{m} = 0$), and M stands for the total number of predicted bounding boxes. λ_{cls} and λ_{box} are hyperparameters that balance the respective contributions of the classification and bounding box regression loss terms.

Moreover, unlike pruning measures for classification tasks, we propose to include location information in the utility calculation. To be more specific, we consider both ground truth bounding box information and contextual information surrounding the bounding boxes. Including the object location information can direct more attention to the foreground objects of interest and help filter out irrelevant distractions. Additionally, surrounding regions provide valuable contextual cues for accurate object detection. Thus, we define our location-weighted utility map as:

$$\widetilde{w^{kl}_{(i,j)}} = \beta^{l}_{(i,j)}w^{kl}_{(i,j)}, \tag{3}$$

where $\beta^{l}_{(i,j)}$ denotes the reweighting coefficient associated with the given location within layer l, which is defined as:

$$\beta^{l}_{(i,j)} = \begin{cases} 1 & if\ (i,j)\ \in area(\mathcal{B}) \\ f_{decay}(i,j) & else\ if\ (i,j)\ \in neighbor(\mathcal{B}) \\ 0 & otherwise \end{cases}, \tag{4}$$

where $\mathcal{B}$ denotes a set of the ground truth bounding boxes, $area(\mathcal{B})$ corresponds to the set of pixels within these bounding boxes, and $neighbor(\mathcal{B})$ represents a set of adjacent pixels surrounding the bounding boxes. The size of these margins is proportional to the dimensions of the bounding boxes and the ratio is determined empirically. Furthermore, $f_{decay}(\cdot,\cdot)$ is a decay function defined as follows:

$$f_{decay}(i,j) = m\{(i-a)^2 + (j-b)^2\}^{-s}, \tag{5}$$

where m represents a nonzero constant, (a, b) stands for the center of the corresponding ground truth bounding box, and s is a real-valued number. In addition to the pixels inside the ground truth box, we take into account the neighboring pixels while adjusting their attention weights. This reweighting is related to the distance of a given neighboring pixel from the bounding box center.

For a given input sample, we define the importance of the kth channel of layer l as follows:

$$S^{kl} = |\sum_{i}\sum_{j}\widetilde{w^{kl}_{(i,j)}}|. \tag{6}$$

The algorithm described in Algorithm 1 outlines the core procedure of our proposed channel pruning method. Unlike conventional pruning techniques that rely on generic importance measures such as magnitude or variance, our method employs a task-specific importance measure that reflects each channel's contribution to the final detection utility. This measure incorporates three key components: detection-related saliency, actual object location information, and surrounding contextual information. Many existing pruning techniques are designed primarily for classification tasks and do not incorporate location information, which is crucial for visual detection. The integration of our

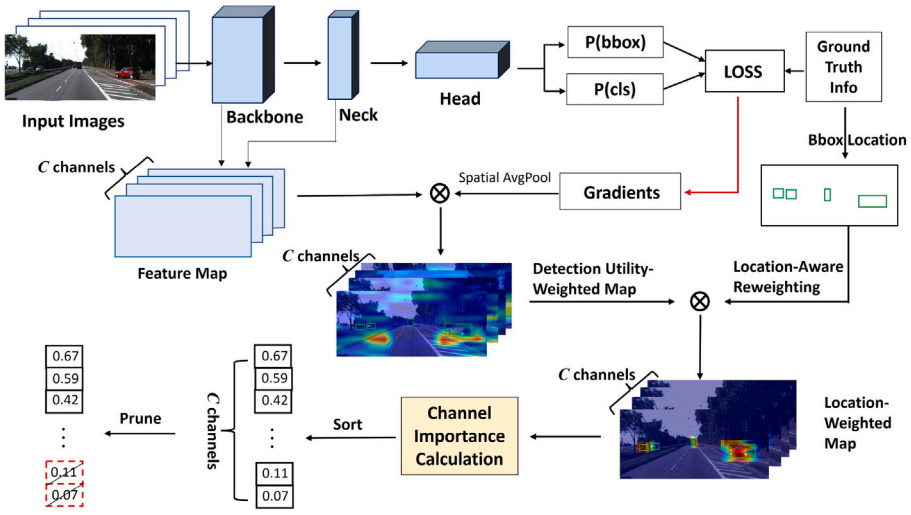


Fig. 1. Overview of the proposed saliency and location aware pruning approach for visual detectors. The location-aware reweighting and the channel importance calculation are based on Eqs. (4) and (6), respectively. C denotes the number of channels in a layer and the dotted red boxes in the figure indicate unimportant channels to be pruned.

key elements ensures that the pruning process preserves features that are vital for accurately identifying and localizing objects in the scene. In Section 4.3.1, we demonstrate that each of these components positively impacts the overall performance of the pruned visual detectors, and together, they contribute significantly to the model's improvement.

Algorithm 1 Saliency & Location Aware Pruning for Layer l

Input: pre-trained model $\mathcal{M}$, pruning rate η, sample size N, layer l with K channels
Output: pruned model $\mathcal{M}'$

1: **for** each sample image **do**
2: **for** each channel k **do**
3: Calculate the detection-utility-weighted feature: $w^{kl}_{(i,j)} = \left(\frac{1}{WH}\sum_{i=1}^{W}\sum_{j=1}^{H}\frac{\partial \mathcal{L}_{det}}{\partial A^{kl}_{(i,j)}}\right)A^{kl}_{(i,j)}$
4: Calculate the location-weighted utility map: $\widetilde{w^{kl}_{(i,j)}} = \beta^{l}_{(i,j)} w^{kl}_{(i,j)}$
5: Calculate the saliency and location aware channel importance: $S^{kl} = |\sum_i \sum_j \widetilde{w^{kl}_{(i,j)}}|$
6: **end for**
7: **end for**
8: Average across N samples: $S^{kl}_{avg} = \frac{\sum_{n=1}^{N} S^{kl}_{n}}{N}$
9: Remove the channels with the lowest S^{kl}_{avg} and their corresponding filters based on the pruning rate η
10: Fine-tune the pruned model

4. Experiments and results

4.1. Experimental setup

In addition to two cutting-edge YOLO family members (i.e., YOLOX-S [21] and YOLOF [22]), we have explored ATSS [19] and GFL [20] as base models in our experiments. CSPDarkNet53 [53] and ResNet50 [54] are employed as the backbone architectures for YOLOX-S and the other three models, respectively. All baseline models have been pre-trained on MS-COCO 2017 [55]. We test our pruning approach on two datasets, i.e., KITTI [56] and COCO_traffic [57]. For KITTI, we have followed the same pre-processing as [58]. Specifically, the KITTI dataset includes three categories: car, cyclist, and pedestrian where the categories of car, van, truck, and tram have been combined into a single category, car. We divide the 7,481 training images into two halves, with the first half used as the training set and the second half reserved for validation (as the testing images lack labels). COCO_traffic [57] is a subset of COCO dataset [55], which is composed of images containing at least one of the 13 traffic-related categories (i.e., person, bicycle, car, motorcycle, bus, train, truck, traffic light, fire hydrant, stop sign, parking meter, dog, and cat). The dataset comprises 25,180 training images and 1,092 testing images in total.

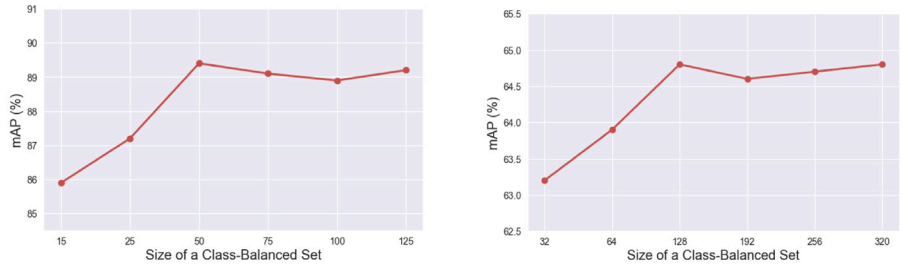
As a data-driven approach, our importance measure is computed based on training data. However, employing the entire dataset for calculating channel importance is generally unfeasible (and unnecessary as we will demonstrate later) for large datasets. For this task, we propose to utilize a class-balanced subset of the training images. To determine the optimal subset size, we conduct experiments with class-balanced subsets of varying sizes, and the pruning results using these subsets can be seen in Fig. 2. The experimentation involves YOLOX-S on both KITTI and COCO_traffic. Notably, for both datasets, the performance reaches a plateau after a certain subset size, beyond which there is minimal improvement. Taking this into account, we use a class-balanced set containing 50 training images for KITTI and 128 training images for COCO_traffic to evaluate the average channel importance.

After completing the channel pruning at a specified pruning rate,[1] we proceed to fine-tune the pruned models using the SGD optimizer. To be specific, we re-trained the pruned model for 300 epochs for YOLOX-S and YOLOF, and 24 epochs for ATSS and GFL, on the KITTI dataset. On COCO_traffic, we performed 100 epochs of re-training for YOLOX-S and YOLOF, and 24 epochs for ATSS and GFL. For both datasets, we use a batch size of 8, a momentum of 0.9, a weight decay of 0.0005, and an initial learning rate of 0.01. The mean Average Precision (mAP) with IoU threshold of 0.5 is used as our overall performance metric. Furthermore, we compute Average Precisions (APs) for various object scales — small, medium, and large — for each dataset. In this context, AP_S, AP_M, and AP_L correspond to the AP scores for objects categorized as small (area less than 32^2), medium (area ranging from 32^2 to 96^2), and large (area exceeding 96^2), respectively.

4.2. Comparative analysis: mAP vs. Pruning rate

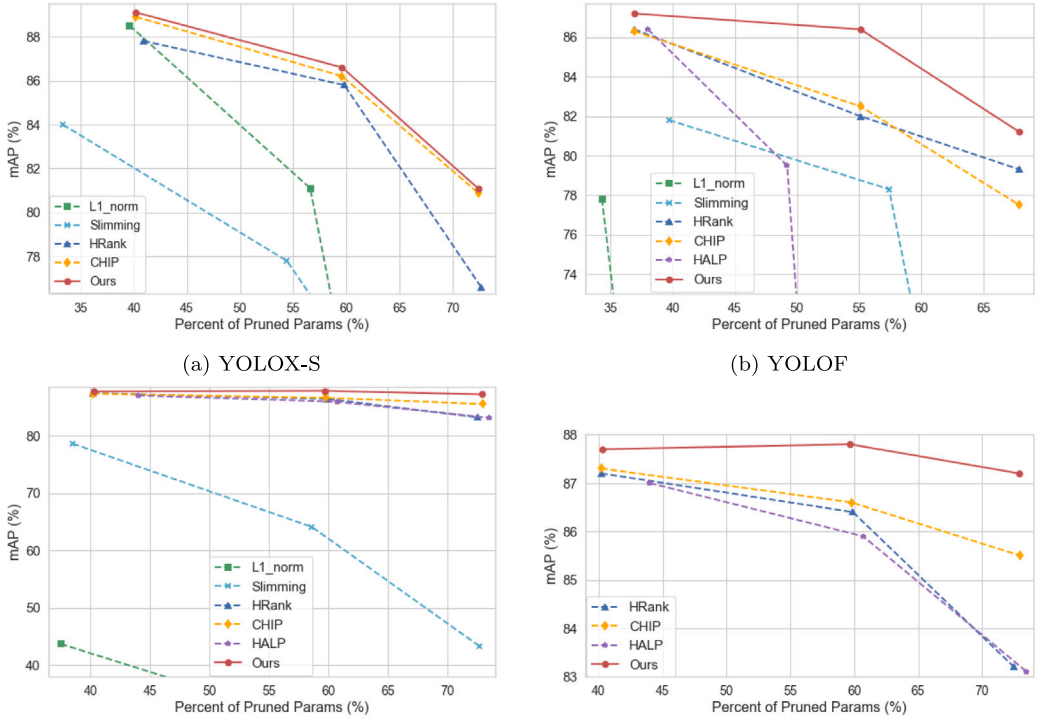
In this subsection, we assess the efficacy of our proposed pruning method on the different base detectors (namely, YOLOX-S, YOLOF,

[1] By default, the pruning rate refers to the channel pruning rate or the percentage of channels discarded in this paper.



(a) KITTI (b) COCO_traffic

Fig. 2. mAP performance of the pruned models vs. training images used during channel importance calculation. Each class is equally represented in the selected training images. Base: YOLOX-S, pruning rate: 30%, datasets: (a) KITTI and (b) COCO_traffic. In both cases, the performance reaches a plateau once a specific subset size is achieved.



(a) YOLOX-S (b) YOLOF

(c) ATSS (d) ATSS_zoomed

Fig. 3. Pruning of (a) YOLOX-S, (b) YOLOF, (c) ATSS on KITTI. (d) is a zoomed-in view of (c). The competing pruning approaches are L1 Pruning [2], Net Slimming [24], HRank [17], CHIP [16], and HALP [23]. The x-axis is the percentage of the parameters that have been discarded, and the y-axis indicates the performance of pruned models.

ATSS, and GFL) at different pruning rates. These evaluations are conducted using both the KITTI and COCO_traffic datasets. Then, we perform a comparative analysis between our approach and a range of classic and/or SOTA pruning methods including, $L1$-norm based pruning [2], Network Slimming [24], HRank [17], CHIP [16], and HALP [23] for a comprehensive comparison.

4.2.1. Results on KITTI

First, on the KITTI dataset, our pruning approach is applied to YOLOX-S, YOLOF, and ATSS models. We then assess its pruning efficacy in comparison to several SOTA pruning methods. As HRank [17] and CHIP [16] did not provide details regarding their specific layer-wise sparsity configurations, we adopt a uniform pruning rate for these two techniques. For a fair comparison, we also adopt this uniform pruning rate for our approach. In [23], only one latency look-up table was provided for ResNets. To ensure a fair comparison, our comparison with HALP [23] was conducted on base detectors using a ResNet backbone, including ATSS, YOLOF, and GFL. The impact of our pruning on model performance is illustrated in Fig. 3. This figure shows the results for YOLOX-S, YOLOF, and ATSS in (a), (b), and (c), respectively. Across all the different pruning rates examined in our experiment, our proposed method consistently outperforms the other approaches. For a closer examination, Fig. 3(d) provides a zoomed-in view of (c). This focused view allows for a more detailed comparison of our method with the three competing approaches: HRank, CHIP, and HALP.

The detailed results for ATSS, YOLOF, and YOLOX-S are shown in Tables 1, 2, and 3, respectively. In Table 1, our approach maintains a comparable performance with a 0.1% mAP reduction, even as parameters are pruned by 59.7% (33.6% reduction in FLOPs). As shown in Table 2, the proposed approach maintains a competitive performance, with only a marginal 0.2% mAP decrease, despite a substantial reduction of 36.9% in parameters (43.1% reduction in FLOPs). Similarly, in Table 3, our approach even achieves an improvement of 1.8% mAP over the original model while pruning 40.2% of the parameters (29.2% reduction in FLOPs). In addition to the overall mAP, our method outperforms the competing SOTA approaches in detecting small-scale objects across a range of pruning rates and baseline models.

Table 1
mAP performance of pruned models (base: ATSS, dataset: KITTI). The numbers in parentheses indicate the percentage reduction compared to the unpruned model. '–' indicates that we did not explore further pruning rates due to excessive loss of important features, with some layers ending up with no filters, breaking the model's information flow.

Approach	Pruning rate (%)[a]	FLOPs (G)	Params (M)	AP_S[b] (%)	AP_M[b] (%)	AP_L[b] (%)	mAP (%)
Original ATSS	0	219.74	32.12	79.1	88.8	93.5	87.9
L1 Pruning [2]	30	165.17 (−24.8%)	20.08 (−37.5%)	39.4	59.5	17.6	43.7
Net Slimming [24]		176.60 (−19.6%)	19.75 (−38.5%)	67.4	83.0	77.6	78.6
HRank [17]		163.69 (−25.5%)	19.21 (−40.2%)	78.9	87.7	93.5	87.4
CHIP [16]		163.67 (−25.5%)	19.20 (−40.2%)	77.5	88.3	93.2	87.3
HALP [23]		178.54 (−18.7%)	17.98 (−44.0%)	78.1	87.8	93.2	87.0
Ours		170.67 (−22.3%)	19.18 (−40.3%)	**79.1**	**88.6**	**93.3**	**87.7**
L1 Pruning [2]	50	148.78 (−32.3%)	13.60 (−57.7%)	31.3	42.6	11.5	30.6
Net Slimming [24]		163.53 (−25.6%)	13.30 (−58.6%)	45.0	66.8	77.3	64.1
HRank [17]		135.22 (−38.5%)	12.87 (−59.9%)	78.3	86.9	92.8	86.4
CHIP [16]		135.30 (−38.4%)	12.91 (−59.8%)	78.2	87.3	92.7	86.6
HALP [23]		150.78 (−31.4%)	12.61 (−60.7%)	77.7	86.4	92.3	85.9
Ours		145.80 (−33.6%)	12.93 (−59.7%)	**79.3**	**88.4**	**93.3**	**87.8**
L1 Pruning [2]	70	143.69 (−34.6%)	9.40 (−70.7%)	28.4	37.1	10.2	26.7
Net Slimming [24]		149.38 (−32.0%)	8.78 (−72.7%)	40.2	58.6	17.9	43.3
HRank [17]		116.24 (−47.1%)	8.83 (−72.5%)	71.8	84.7	91.5	83.2
CHIP [16]		115.46 (−47.5%)	8.68 (−73.0%)	75.1	86.5	92.8	85.5
HALP [23]		134.43 (−38.8%)	8.52 (−73.5%)	72.0	84.9	90.2	83.1
Ours		127.78 (−41.8%)	8.71 (−72.9%)	**78.3**	**88.0**	**92.9**	**87.2**
L1 Pruning [2]	80	127.29 (−42.1%)	7.96 (−75.2%)	<10	<10	<10	<10
Net Slimming [24]		126.97 (−42.2%)	7.70 (−76.0%)	<10	<10	<10	<10
HRank [17]		101.88 (−53.6%)	6.00 (−81.3%)	69.8	82.9	88.7	80.9
CHIP [16]		100.85 (−54.1%)	5.83 (−81.8%)	69.0	83.3	89.4	81.4
HALP [23]		–	–	–	–	–	–
Ours		101.52 (−53.8%)	5.93 (−81.5%)	**71.1**	**84.4**	**89.6**	**82.5**
L1 Pruning [2]	90	–	–	–	–	–	–
Net Slimming [24]		–	–	–	–	–	–
HRank [17]		86.70 (−60.5%)	3.64 (−88.7%)	33.8	50.9	62.7	49.4
CHIP [16]		86.05 (−60.8%)	3.59 (−88.8%)	34.1	48.3	58.1	47.1
HALP [23]		–	–	–	–	–	–
Ours		86.43 (−60.7%)	3.61 (−88.8%)	**39.7**	**58.0**	**64.7**	**54.9**

[a] Pruning rate refers to the channel pruning rate or the percentage of channels discarded.

[b] AP_S, AP_M, and AP_L indicate the Average Precision scores for small, medium, and large objects, respectively.

4.2.2. Results on COCO_traffic

We assess the effectiveness of our approach on the COCO_traffic dataset using YOLOX-S, YOLOF, and GFL models and compare it with the competing methods. The results for these detectors are illustrated in Fig. 4. Across all pruning rates explored in our study, our approach consistently achieves the highest mAP among the pruning methods for all three base models. Similar to Fig. 3, for a more detailed analysis, Fig. 4(d) offers a zoomed-in view of (c).

The detailed results for GFL, YOLOF, and YOLOX-S models are shown in Tables 4, 5, and 6, respectively. On the COCO_traffic dataset, our approach also demonstrates superior performance in the detection of small-scale objects when compared to the competing SOTA approaches. For example, our pruned GFL, YOLOF, and YOLOX-S models can achieve an average of 7.6%, 6.3%, and 7.5% higher mAP for small-scale objects compared to models using the other competitive methods, with 40.1%, 36.9%, and 50.7% model size reductions, respectively.

In Fig. 5, we visually compare our method's detection results with those of the competing pruning approaches on COCO_traffic examples. As we can see, our approach demonstrates the best detection performance, especially for small-scale cases (as denoted by cyan circles in Fig. 5). A potential explanation is that bounding boxes of smaller objects inherently offer limited information, causing the surrounding areas to assume a more significant and crucial role in such cases. Our approach can better take advantage of such neighboring contextual information in addition to the bounding box information.

4.3. Ablation analysis

To examine and assess the effects of various factors on our pruned model's performance, we perform a comprehensive ablation analysis. This study evaluates the influence of individual importance components and decay weighting functions on our detection utility measure.

4.3.1. Impact of individual importance elements

We investigate the three elements of our saliency-based, location-aware channel importance measure: the gradients of the detection utility, ground truth bounding boxes, and the contextual information in the vicinity. We remove one element at a time, conduct the channel pruning, and assess its effects on channel pruning performance. The experiments are conducted on the KITTI dataset, using YOLOX-S as the base model. The pruning rate is set at 30% for all experiments. The outcomes are presented in Table 7. Notably, each element contributes positively to the overall performance of our pruned model, and the model, which takes into account all three elements when computing channel importance, attains the highest mAP. This illustrates the effectiveness of the proposed method, which integrates these elements to optimize channel pruning.

4.3.2. Decay weighting functions

In our approach, we enhance the ground truth bounding boxes by incorporating additional contextual information. This involves introducing margins to the boxes and applying a decay function to the surrounding regions. The importance of the surrounding context in boosting the performance of the pruned model has been previously testified. Here, we assess the effects of substituting decay functions on model performance. For our analysis, we focus on the YOLOX-S model and conduct experiments on the KITTI dataset at various parameter pruning rates (i.e., 40.2%, 59.6%, and 72.4%, corresponding to 30%,

Table 2
mAP performance of pruned models (base: YOLOF, dataset: KITTI). The numbers in parentheses indicate the percentage reduction compared to the unpruned model. '–' indicates that we did not explore further pruning rates due to excessive loss of important features, with some layers ending up with no filters, breaking the model's information flow.

Approach	Pruning rate (%)[a]	FLOPs (G)	Params (M)	AP_S[b] (%)	AP_M[b] (%)	AP_L[b] (%)	mAP (%)
Original YOLOF	0	106.90	42.39	74.4	89.4	93.6	87.4
L1 Pruning [2]	30	63.09 (−41.0%)	27.87 (−34.3%)	65.1	80.1	84.0	77.8
Net Slimming [24]		80.54 (−24.1%)	25.54 (−39.7%)	67.3	84.5	89.4	81.8
HRank [17]		60.48 (−43.4%)	25.59 (−37.3%)	74.8	88.2	92.6	86.4
CHIP [16]		60.83 (−43.1%)	26.75 (−36.9%)	74.1	88.2	92.4	86.3
HALP [23]		55.47 (−48.1%)	26.28 (−38.0%)	71.8	89.1	94.1	86.4
Ours		60.83 (−43.1%)	26.75 (−36.9%)	**76.6**	**88.8**	**92.7**	**87.2**
L1 Pruning [2]	50	50.83 (−52.5%)	22.34 (−47.3%)	<10	<10	<10	<10
Net Slimming [24]		45.89 (−57.1%)	18.04 (−57.4%)	62.1	81.7	85.9	78.3
HRank [17]		38.07 (−64.4%)	18.90 (−55.4%)	66.4	84.9	90.3	82.0
CHIP [16]		38.34 (−64.1%)	19.02 (−55.1%)	67.7	85.2	90.3	82.5
HALP [23]		39.12 (−63.4%)	21.52 (−49.2%)	61.3	83.2	89.6	79.5
Ours		38.34 (−64.1%)	19.02 (−55.1%)	**74.2**	**88.4**	**92.3**	**86.4**
L1 Pruning [2]	70	39.77 (−62.8%)	17.32 (−59.1%)	<10	<10	<10	<10
Net Slimming [24]		27.29 (−74.5%)	15.22 (−64.1%)	41.2	59.9	65.3	57.4
HRank [17]		22.88 (−78.6%)	13.52 (−68.1%)	63.4	82.3	87.7	79.3
CHIP [16]		23.23 (−78.3%)	13.61 (−67.9%)	61.5	80.4	86.3	77.5
HALP [23]		24.44 (−77.1%)	18.52 (−56.3%)	<10	19.7	18.9	17.0
Ours		23.23 (−78.3%)	13.61 (−67.9%)	**64.7**	**84.4**	**89.3**	**81.2**
L1 Pruning [2]	80	–	–	–	–	–	–
Net Slimming [24]		13.35 (−87.5%)	9.75 (−77.0%)	<10	<10	<10	<10
HRank [17]		13.37 (−87.5%)	9.74 (−77.0%)	45.7	65.6	71.6	62.8
CHIP [16]		13.32 (−87.5%)	9.72 (−77.1%)	48.6	69.4	77.5	66.8
HALP [23]		–	–	–	–	–	–
Ours		13.35 (−87.5%)	9.73 (−77.0%)	**51.5**	**71.1**	**80.1**	**68.8**

[a] Pruning rate refers to the channel pruning rate or the percentage of channels discarded.
[b] AP_S, AP_M, and AP_L indicate the Average Precision scores for small, medium, and large objects, respectively.

Table 3
mAP performance of pruned models (base: YOLOX-S, dataset: KITTI). The numbers in parentheses indicate the percentage reduction compared to the unpruned model.

Approach	Pruning rate (%)[a]	FLOPs (G)	Params (M)	AP_S[b] (%)	AP_M[b] (%)	AP_L[b] (%)	mAP (%)
Original YOLOX-S	0	26.65	8.94	75.4	89.4	92.8	87.3
L1 Pruning [2]	30	18.71 (−29.8%)	5.40 (−39.6%)	78.7	90.0	93.4	88.5
Net Slimming [24]		16.49 (−38.1%)	5.96 (−33.3%)	68.3	86.3	93.1	84.0
HRank [17]		18.76 (−29.6%)	5.28 (−40.9%)	77.2	89.6	92.3	87.8
CHIP [16]		18.88 (−29.2%)	5.35 (−40.2%)	79.9	90.3	92.7	88.9
Ours		18.88 (−29.2%)	5.35 (−40.2%)	**81.1**	**90.1**	**93.1**	**89.1**
L1 Pruning [2]	50	14.99 (−43.8%)	3.88 (−56.6%)	65.9	83.1	89.6	81.1
Net Slimming [24]		14.22 (−46.6%)	4.08 (−54.4%)	57.8	80.0	91.2	77.8
HRank [17]		14.99 (−43.8%)	3.59 (−59.8%)	71.4	87.1	92.3	85.2
CHIP [16]		15.01 (−43.7%)	3.61 (−59.6%)	74.1	88.1	92.3	86.2
Ours		15.01 (−43.7%)	3.61 (−59.6%)	**75.1**	**88.1**	**92.6**	**86.6**
L1 Pruning [2]	70	12.36 (−53.6%)	2.22 (−75.2%)	24.1	33.6	48.6	34.9
Net Slimming [24]		12.69(−52.4%)	2.66 (−70.2%)	44.7	67.9	84.7	66.7
HRank [17]		12.37 (−53.6%)	2.44 (−72.7%)	57.2	78.4	90.0	76.6
CHIP [16]		12.50 (−53.1%)	2.47 (−72.4%)	64.0	83.2	91.4	80.9
Ours		12.50 (−53.1%)	2.47 (−72.4%)	**64.4**	**83.1**	**91.6**	**81.1**
L1 Pruning [2]	80	11.93 (−55.2%)	1.87 (−79.1%)	21.3	30.0	42.6	31.2
Net Slimming [24]		11.74 (−55.9%)	1.88 (−79.0%)	26.0	44.3	65.2	45.1
HRank [17]		10.85 (−59.3%)	1.83 (−79.5%)	33.9	58.3	70.2	55.5
CHIP [16]		10.81 (−59.4%)	1.82 (−79.6%)	36.5	59.8	75.2	57.9
Ours		10.81 (−59.4%)	1.82 (−79.6%)	**37.4**	**62.0**	**77.1**	**59.8**

[a] Pruning rate refers to the channel pruning rate or the percentage of channels discarded.
[b] AP_S, AP_M, and AP_L indicate the Average Precision scores for small, medium, and large objects, respectively.

50%, and 70% of channels discarded). We investigate three different decay functions for the reweighting purpose: the flat-top Gaussian function, the exponential function, and the power function. The results are visually presented in Fig. 6. Among the evaluated decay functions, the model using a power decay function demonstrates superior performance. On the other hand, the model produced without any decay function exhibits the poorest results, highlighting the importance of incorporating a decay function to effectively leverage contextual cues when calculating channel importance. Given the better performance of the power decay function, we have selected it as our preferred decay function (as shown in Eq. (5)).

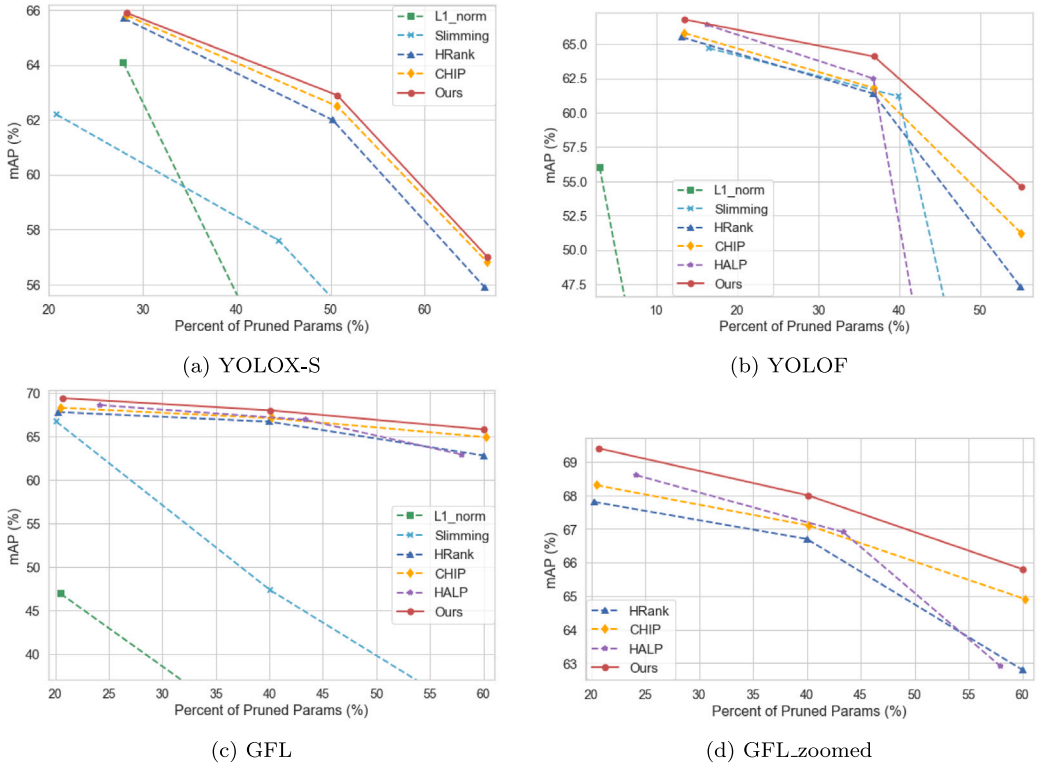


Fig. 4. Pruning of (a) YOLOX-S, (b) YOLOF, (c) GFL on COCO_traffic. (d) is a zoomed-in view of (c). The competing pruning approaches are L1 Pruning [2], Net Slimming [24], HRank [17], CHIP [16], and HALP [23]. The x-axis is the percentage of the parameters that have been discarded, and the y-axis indicates the performance of pruned models.

Table 4
mAP performance of pruned models (base: GFL, dataset: COCO_traffic). The numbers in parentheses indicate the percentage reduction compared to the unpruned model.

Approach	Pruning rate (%)[a]	FLOPs (G)	Params (M)	AP_S[b] (%)	AP_M[b] (%)	AP_L[b] (%)	mAP (%)
Original GFL	0	223.66	32.29	46.2	71.7	89.1	71.5
L1 Pruning [2]	20	189.90 (−15.1%)	25.66 (−20.5%)	21.5	47.5	70.2	46.9
Net Slimming [24]		196.75 (−12.0%)	25.79 (−20.1%)	39.2	67.1	85.9	66.7
HRank [17]		195.44 (−12.6%)	25.73 (−20.3%)	44.8	70.8	85.4	67.8
CHIP [16]		195.30 (−12.7%)	25.66 (−20.5%)	43.2	69.5	87.0	68.3
HALP [23]		200.43 (−10.4%)	24.48 (−24.2%)	46.0	70.9	85.5	68.6
Ours		198.66 (−11.2%)	25.60 (−20.7%)	**46.5**	**70.3**	**86.3**	**69.4**
L1 Pruning [2]	40	181.07 (−19.0%)	20.39 (−36.9%)	15.6	29.6	51.1	32.5
Net Slimming [24]		180.35 (−19.4%)	19.33 (−40.1%)	34.3	43.0	63.0	47.3
HRank [17]		167.59 (−25.1%)	19.37 (−40.0%)	41.7	67.1	85.3	66.7
CHIP [16]		167.49 (−25.1%)	19.32 (−40.2%)	46.3	69.8	85.0	67.1
HALP [23]		176.83 (−20.9%)	18.26 (−43.4%)	41.9	68.9	84.4	66.9
Ours		174.59 (−21.9%)	19.35 (−40.1%)	**43.6**	**69.1**	**85.1**	**68.0**
L1 Pruning [2]	60	154.43 (−31.0%)	14.77 (−54.3%)	<10	<10	<10	<10
Net Slimming [24]		164.14 (−26.6%)	12.94 (−59.9%)	15.1	29.4	51.4	32.3
HRank [17]		138.67 (−38.0%)	12.91 (−60.0%)	38.6	64.3	81.9	62.8
CHIP [16]		138.18 (−38.2%)	12.83 (−60.3%)	41.2	68.4	81.6	64.9
HALP [23]		154.40 (−31.0%)	13.57 (−58.0%)	36.9	64.9	80.6	62.9
Ours		148.79 (−33.5%)	12.90 (−60.0%)	**41.8**	**68.4**	**80.5**	**65.8**

[a] Pruning rate refers to the channel pruning rate or the percentage of channels discarded.
[b] AP_S, AP_M, and AP_L indicate the Average Precision scores for small, medium, and large objects, respectively.

4.4. Inference latency analysis

In this section, we explore the practical implications of parameter and FLOP reductions through pruning on inference latency. As an example, we provide insights into the inference latency of the YOLOF model and our pruned variant with a comparable mAP on the KITTI dataset. It is important to note that the direct measurement of inference latency is influenced by various hardware-specific and environmental

Table 5
mAP performance of pruned models (base: YOLOF, dataset: COCO_traffic). The numbers in parentheses indicate the percentage reduction compared to the unpruned model. '–' indicates that we did not explore further pruning rates due to excessive loss of important features, with some layers ending up with no filters, breaking the model's information flow.

Approach	Pruning rate (%)[a]	FLOPs (G)	Params (M)	AP_S[b] (%)	AP_M[b] (%)	AP_L[b] (%)	mAP (%)
Original YOLOF	0	107.15	42.39	42.9	71.0	84.5	67.0
L1 Pruning [2]	10	86.56 (−19.2%)	41.11 (−3.0%)	23.1	60.4	79.4	56.0
Net Slimming [24]		94.83 (−11.5%)	35.40 (−16.5%)	38.7	68.1	82.7	64.7
HRank [17]		87.17 (−18.6%)	36.80 (−13.2%)	37.7	69.3	84.2	65.5
CHIP [16]		90.40 (−15.6%)	36.65 (−13.5%)	38.9	69.7	83.7	65.8
HALP [23]		94.19 (−12.1%)	35.46 (−16.3%)	45.4	72.2	82.9	66.4
Ours		90.40 (−15.6%)	36.65 (−13.5%)	**44.6**	**71.2**	**83.5**	**66.8**
L1 Pruning [2]	30	66.39 (−38.0%)	34.62 (−18.3%)	<10	<10	<10	<10
Net Slimming [24]		57.17 (−46.6%)	25.49 (−39.9%)	34.5	65.3	79.7	61.2
HRank [17]		90.40 (−46.1%)	36.65 (−36.8%)	35.8	64.3	80.8	61.4
CHIP [16]		61.09 (−43.0%)	26.75 (−36.9%)	35.5	66.2	81.5	61.8
HALP [23]		57.60 (−46.2%)	26.81 (−36.8%)	37.9	67.9	78.6	62.5
Ours		61.09 (−43.0%)	26.75 (−36.9%)	**36.8**	**69.8**	**82.3**	**64.1**
L1 Pruning [2]	50	43.23 (−59.7%)	20.58 (−51.5%)	<10	<10	<10	<10
Net Slimming [24]		44.77 (−58.2%)	18.83 (−55.6%)	<10	27.1	25.2	20.2
HRank [17]		35.49 (−66.9%)	19.09 (−55.0%)	20.0	49.0	66.2	47.3
CHIP [16]		38.59 (−64.0%)	19.02 (−55.1%)	26.4	54.7	69.1	51.2
HALP [23]		–	–	–	–	–	–
Ours		38.59 (−64.0%)	19.02 (−55.1%)	**27.8**	**60.8**	**74.4**	**54.6**

[a] Pruning rate refers to the channel pruning rate or the percentage of channels discarded.
[b] AP_S, AP_M, and AP_L indicate the Average Precision scores for small, medium, and large objects, respectively.

Table 6
mAP performance of pruned models (base: YOLOX-S, dataset: COCO_traffic). The numbers in parentheses indicate the percentage reduction compared to the unpruned model.

Approach	Pruning rate (%)[a]	FLOPs (G)	Params (M)	AP_S[b] (%)	AP_M[b] (%)	AP_L[b] (%)	mAP (%)
Original YOLOX-S	0	26.67	8.94	44.6	72.5	83.9	68.7
L1 Pruning [2]	20	20.94 (−21.5%)	6.45 (−27.9%)	43.5	70.9	79.3	64.1
Net Slimming [24]		19.48 (−27.0%)	7.08 (−20.8%)	38.1	67.4	79.6	62.2
HRank [17]		21.31 (−20.1%)	6.44 (−28.0%)	42.9	70.3	81.1	65.7
CHIP [16]		21.29 (−20.2%)	6.41 (−28.3%)	41.2	71.9	79.9	65.8
Ours		21.29 (−20.2%)	6.41 (−28.3%)	**46.0**	**69.2**	**80.6**	**65.9**
L1 Pruning [2]	40	16.48 (−38.2%)	4.78 (−46.5%)	25.9	56.1	68.2	51.1
Net Slimming [24]		16.00 (−40.0%)	4.96 (−44.5%)	30.6	63.5	5.9	57.6
HRank [17]		16.94 (−36.5%)	4.45 (−50.2%)	40.5	67.3	78.3	62.0
CHIP [16]		16.87 (−36.7%)	4.41 (−50.7%)	37.5	66.7	78.4	62.5
Ours		16.87 (−36.7%)	4.41 (−50.7%)	**41.1**	**67.5**	**78.1**	**62.9**
L1 Pruning [2]	60	12.64 (−52.6%)	2.56 (−71.4%)	<10	15.9	30.1	18.4
Net Slimming [24]		13.90 (−47.9%)	3.37 (−62.3%)	25.2	54.5	73.2	51.0
HRank [17]		13.66 (−48.8%)	3.00 (−66.4%)	34.3	61.4	73.7	55.9
CHIP [16]		13.64 (−48.9%)	2.98 (−66.7%)	31.5	62.6	74.7	56.8
Ours		13.64 (−48.9%)	2.98 (−66.7%)	**32.0**	**61.0**	**75.6**	**57.0**

[a] Pruning rate refers to the channel pruning rate or the percentage of channels discarded.
[b] AP_S, AP_M, and AP_L indicate the Average Precision scores for small, medium, and large objects, respectively.

Table 7
Impact of the components of our approach on pruning performance. GT bbox denotes Ground Truth bounding boxes. Base: YOLOX-S, Dataset: KITTI, pruning rate: 30%.

Modules	Gradients	×	✓	✓	✓
	GT bbox	×	×	✓	✓
	Surrounding	×	×	×	✓
Results	mAP	87.3	88.5	88.7	**89.1**

factors. Our experiments were conducted on a single Intel Xeon E5-2680 v4 CPU and a single NVIDIA Tesla P100 GPU. In Table 8, we present the latency results on both the CPU and the GPU, along with the reductions in parameters and FLOPs, as well as the mAP for both the pruned and unpruned models. As detailed in Table 8, the pruned model with 43.1% and 36.9% reductions in FLOPs and parameters, yields substantial improvements in latency, with reductions of 25.5% and 28.3% on the CPU and the GPU, respectively.

5. Limitations and future work

In real-world autonomous driving scenarios, ensuring real-time performance on resource-constrained hardware is critical. While our experiments on the KITTI and COCO_traffic datasets show promising

Fig. 5. Qualitative results of competing SOTA approaches (i.e., L1-norm pruning [2], Net Slimming [24], HRank [17], and CHIP [16]) and our method on example images from COCO_traffic. In each case, the detection image was produced using YOLOX-S pruned at a 40% pruning rate on COCO_traffic. The final row highlights cyan circles, indicated by arrows, demonstrating the superior performance of the detector pruned using our approach over other competing methods. Best viewed when zoomed in.

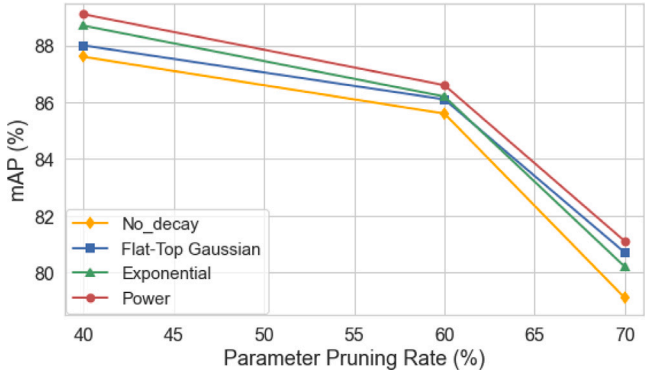


Fig. 6. Pruning performance with various decay functions to capture context information (as in Eq. (4)), across a range of parameter pruning rates. Base: YOLOX-S, dataset: KITTI.

Table 8
Inference latency improvement via our pruning. Base: YOLOF, pruning rate: 30%, dataset: KITTI, hardware specifications: Intel Xeon E5-2680 v4 CPU and NVIDIA Tesla P100 GPU. Each latency number is an average of 100 runs.

Model	FLOPs (G)	Params (M)	CPU latency (ms)	GPU latency (ms)	mAP (%)
Original	106.90	42.39	7250	25.8	87.4
Pruned	**60.83 (−43.1%)**	**26.75 (−36.9%)**	**5400 (−25.5%)**	**18.5 (−28.3%)**	**87.2**

results, further optimizations and hardware-specific implementations are needed to achieve the required speed and efficiency on various embedded systems. We will explore other complementary model compression techniques, such as quantization and knowledge distillation, to further boost model efficiency.

While inductive biases such as spatial locality are reasonable for visual tasks, it would be intriguing to investigate long-range dependencies between objects and background cues, extending beyond intermediate neighborhood contexts. To this end, we plan to modify/extend this work to pruning tokens in the self-attention mechanisms found in transformers.

In addition, our experiments have primarily focused on standard datasets representing common traffic scenarios. However, autonomous vehicles may encounter diverse and dynamic environments, including

varying weather conditions, lighting changes, and unexpected obstacles. The performance of our pruned models in such scenarios has not yet been extensively tested. We will analyze the effect of our pruning on detector robustness, which is a critical concern in safety-critical applications like autonomous driving. We will also design strategies to improve pruned detectors' robustness against adversarial perturbations, thereby contributing to safer and more trustworthy autonomous driving systems.

6. Conclusion

In this paper, we aim to address the growing demand for more efficient deep detectors in the resource-constrained and time-sensitive application of autonomous driving perception. We presented a novel saliency and location aware channel importance measure specifically designed for self-driving visual detection. By integrating relevant object location and contextual details with detection-oriented saliency, our pruning achieved great compression rates while maintaining competitive mAP scores. Through extensive experiments on the KITTI and COCO_traffic datasets using a range of visual detectors including YOLOX-S, YOLOF, ATSS, and GFL, we showed our method's effectiveness and superiority over existing SOTA pruning methods, as well as its potential in dealing with small-scale objects. One of our pruned models outperforms its original model by 1.8% mAP while utilizing only 59.8% of the parameters. This work not only introduces a novel pruning method for visual detectors but also contributes to a better understanding of the challenges of visual detector pruning, which involves both object classification and localization.

CRediT authorship contribution statement

Jung Im Choi: Conceptualization, Formal analysis, Investigation, Methodology, Data curation, Validation, Visualization, Writing – original draft, Writing – review & editing. **Qing Tian:** Conceptualization, Formal analysis, Investigation, Methodology, Writing – original draft, Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

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

Data availability

The data that has been used is publically accessible.

Acknowledgments

This research was supported by the National Science Foundation (NSF), USA under Award No. 2153404 and No. 2412285. This work would not have been possible without the computing resources provided by the Ohio Supercomputer Center. The majority of the work was completed while the PI (Tian) was at Bowling Green State University.

References

[1] S. Han, J. Pool, J. Tran, W.J. Dally, Learning both weights and connections for efficient neural networks, in: Neural Information Processing Systems, NeurIPS, 2015, pp. 1135–1143, http://dx.doi.org/10.5555/2969239.2969366.

[2] H. Li, A. Kadav, I. Durdanovic, H. Samet, H.P. Graf, Pruning filters for efficient convnets, in: International Conference on Learning Representations, ICLR, 2017, http://dx.doi.org/10.48550/arXiv.1608.08710.

[3] Q. Tian, T. Arbel, J.J. Clark, Task dependent deep lda pruning of neural networks, Comput. Vis. Image Underst. 203 (2021) 103154, http://dx.doi.org/10.1016/j.cviu.2020.103154.

[4] G.E. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, 2015, CoRR abs/1503.02531. URL http://arxiv.org/abs/1503.02531.

[5] P. Chen, S. Liu, H. Zhao, J. Jia, Distilling knowledge via knowledge review, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2021, pp. 5006–5015, http://dx.doi.org/10.1109/CVPR46437.2021.00497.

[6] W. Chen, J. Wilson, S. Tyree, K. Weinberger, Y. Chen, Compressing neural networks with the hashing trick, in: International Conference on Machine Learning, ICML, PMLR, 2015, pp. 2285–2294, URL https://proceedings.mlr.press/v37/chenc15.html.

[7] A. Polino, R. Pascanu, D. Alistarh, Model compression via distillation and quantization, 2018, CoRR abs/1802.05668. arXiv:1802.05668. URL http://arxiv.org/abs/1802.05668.

[8] Y. LeCun, J.S. Denker, S.A. Solla, Optimal brain damage, in: Neural Information Processing Systems, NeurIPS, 1989, pp. 598–605, URL https://api.semanticscholar.org/CorpusID:7785881.

[9] B. Hassibi, D.G. Stork, Second order derivatives for network pruning: Optimal brain surgeon, in: Neural Information Processing Systems, NeurIPS, 1992, pp. 164–171, http://dx.doi.org/10.5555/2987061.2987082.

[10] H. Hu, R. Peng, Y.-W. Tai, C.-K. Tang, Network trimming: A data-driven neuron pruning approach towards efficient deep architectures, 2016, pp. 1–9, http://dx.doi.org/10.48550/arXiv.1607.03250, CoRR abs/1607.03250.

[11] P. Molchanov, S. Tyree, T. Karras, T. Aila, J. Kautz, Pruning convolutional neural networks for resource efficient inference, in: International Conference on Learning Representations, ICLR, 2017, http://dx.doi.org/10.48550/arXiv.1611.06440.

[12] Q. Tian, T. Arbel, J.J. Clark, Deep lda-pruned nets for efficient facial gender classification, in: IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPRW, 2017, pp. 512–521, http://dx.doi.org/10.1109/CVPRW.2017.78.

[13] Y. He, G. Kang, X. Dong, Y. Fu, Y. Yang, Soft filter pruning for accelerating deep convolutional neural networks, in: International Joint Conference on Artificial Intelligence, IJCAI, 2018, pp. 2234–2240, http://dx.doi.org/10.5555/3304889.3304970.

[14] Y. He, P. Liu, Z. Wang, Z. Hu, Y. Yang, Filter pruning via geometric median for deep convolutional neural networks acceleration, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2019, pp. 4335–4344, http://dx.doi.org/10.1109/CVPR.2019.00447.

[15] Y. Guo, A. Yao, Y. Chen, Dynamic network surgery for efficient dnns, in: Neural Information Processing Systems, NeurIPS, 2016, pp. 1379–1387, http://dx.doi.org/10.5555/3157096.3157251.

[16] Y. Sui, M. Yin, Y. Xie, H. Phan, S. Zonouz, B. Yuan, Chip: Channel independence-based pruning for compact neural networks, in: Neural Information Processing Systems, NeurIPS, Vol. 34, 2021, pp. 24604–24616, URL https://proceedings.neurips.cc/paper_files/paper/2021/file/ce6babd060aa46c61a5777902cca78af-Paper.pdf.

[17] M. Lin, R. Ji, Y. Wang, Y. Zhang, B. Zhang, Y. Tian, L. Shao, Hrank: Filter pruning using high-rank feature map, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2020, pp. 1526–1535, http://dx.doi.org/10.1109/CVPR42600.2020.00160.

[18] J.I. Choi, Q. Tian, Visual-saliency-guided channel pruning for deep visual detectors in autonomous driving, in: IEEE Intelligent Vehicles Symposium, IV, 2023, http://dx.doi.org/10.1109/IV55152.2023.10186819.

[19] S. Zhang, C. Chi, Y. Yao, Z. Lei, S.Z. Li, Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2020, pp. 9756–9765, http://dx.doi.org/10.1109/CVPR42600.2020.00978.

[20] X. Li, W. Wang, L. Wu, S. Chen, X. Hu, J. Li, J. Tang, J. Yang, Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection, in: Neural Information Processing Systems, NeurIPS, 2020.

[21] Z. Ge, S. Liu, F. Wang, Z. Li, J. Sun, YOLOX: exceeding YOLO series in 2021, 2021, CoRR abs/2107.08430. URL https://arxiv.org/abs/2107.08430.

[22] Q. Chen, Y. Wang, T. Yang, X. Zhang, J. Cheng, J. Sun, You only look one-level feature, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2021, pp. 13039–13048, http://dx.doi.org/10.1109/CVPR46437.2021.01284.

[23] M. Shen, H. Yin, P. Molchanov, L. Mao, J. Liu, J.M. Alvarez, Structural pruning via latency-saliency knapsack, in: Advances in Neural Information Processing Systems, Vol. 35, Curran Associates, Inc., 2022, pp. 12894–12908.

[24] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, C. Zhang, Learning efficient convolutional networks through network slimming, in: IEEE International Conference on Computer Vision, ICCV, 2017, pp. 2755–2763, http://dx.doi.org/10.1109/ICCV.2017.298.

[25] R. Vaillant, C. Monrocq, Y. Le Cun, Original approach for the localization of objects in images, IEE Proc.: Vis. Image Signal Process. 141 (4) (1994) 245–250, http://dx.doi.org/10.1049/ip-vis:19941301.

[26] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Vol. 1, 2001, p. I, http://dx.doi.org/10.1109/CVPR.2001.990517.

[27] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, in: Advances in Neural Information Processing Systems, NeurIPS, Vol. 28, 2015, pp. 91–99, http://dx.doi.org/10.5555/2969239.2969250.

[28] A. Bochkovskiy, C.-Y. Wang, H.-Y.M. Liao, Yolov4: Optimal speed and accuracy of object detection, 2020, http://dx.doi.org/10.48550/arXiv.2004.10934, arXiv preprint arXiv:2004.10934.
[29] H. Fujiyoshi, T. Hirakawa, T. Yamashita, Deep learning-based image recognition for autonomous driving, IATSS Res. 43 (4) (2019) 244–252, http://dx.doi.org/10.1016/j.iatssr.2019.11.008.
[30] S. Grigorescu, B. Trasnea, T. Cocias, G. Macesanu, A survey of deep learning techniques for autonomous driving, J. Field Robotics 37 (3) (2020) 362–386, http://dx.doi.org/10.1002/rob.21918.
[31] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2014, pp. 580–587, http://dx.doi.org/10.1109/CVPR.2014.81.
[32] R. Girshick, Fast r-cnn, in: IEEE International Conference on Computer Vision, ICCV, 2015, pp. 1440–1448, http://dx.doi.org/10.1109/ICCV.2015.169.
[33] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2016, pp. 779–788, http://dx.doi.org/10.1109/CVPR.2016.91.
[34] T. Zhang, S. Ye, K. Zhang, J. Tang, W. Wen, M. Fardad, Y. Wang, A systematic dnn weight pruning framework using alternating direction method of multipliers, in: European Conference Computer Vision, ECCV, 2018, pp. 191–207, http://dx.doi.org/10.1007/978-3-030-01237-3_12.
[35] S. Han, H. Mao, W.J. Dally, Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, in: International Conference on Learning Representations, ICLR, 2016, http://dx.doi.org/10.48550/arXiv.1510.00149.
[36] S. Park, J. Lee, S. Mo, J. Shin, Lookahead: a far-sighted alternative of magnitude-based pruning, 2020, http://dx.doi.org/10.48550/arXiv.2002.04809, CoRR abs/2002.04809. arXiv:2002.04809.
[37] J. Park, S.R. Li, W. Wen, P.T.P. Tang, H.H. Li, Y. Chen, P.K. Dubey, Faster cnns with direct sparse convolutions and guided pruning, in: International Conference of Learning Representation, ICLR, 2017, http://dx.doi.org/10.48550/arXiv.1608.01409.
[38] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. Horowitz, W.J. Dally, Eie: Efficient inference engine on compressed deep neural network, in: International Symposium on Computer Architecture, ISCA, 2016, pp. 243–254, http://dx.doi.org/10.1109/ISCA.2016.30.
[39] C. Liu, H. Wu, Channel pruning based on mean gradient for accelerating convolutional neural networks, Signal Process. 156 (2019) 84–91, http://dx.doi.org/10.1016/j.sigpro.2018.10.019, URL https://www.sciencedirect.com/science/article/pii/S0165168418303517.
[40] Y. He, X. Zhang, J. Sun, Channel pruning for accelerating very deep neural networks, in: IEEE International Conference on Computer Vision, ICCV, 2017, pp. 1398–1406, http://dx.doi.org/10.1109/ICCV.2017.155.
[41] J.-H. Luo, J. Wu, W. Lin, Thinet: A filter level pruning method for deep neural network compression, in: IEEE International Conference on Computer Vision, ICCV, 2017, pp. 5068–5076, http://dx.doi.org/10.1109/ICCV.2017.541.
[42] Y. Huang, N. Liu, Z. Che, Z. Xu, C. Shen, Y. Peng, G. Zhang, X. Liu, F. Feng, J. Tang, Cp3: Channel pruning plug-in for point-based networks, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2023, pp. 5302–5312, http://dx.doi.org/10.1109/CVPR52729.2023.00513.
[43] S. Guo, L. Zhang, X. Zheng, Y. Wang, Y. Li, F. Chao, C. Wu, S. Zhang, R. Ji, Automatic network pruning via hilbert-schmidt independence criterion lasso under information bottleneck principle, in: IEEE International Conference on Computer Vision, ICCV, 2023, URL https://openaccess.thecvf.com/content/ICCV2023/papers/Guo_Automatic_Network_Pruning_via_Hilbert--Schmidt_Independence_Criterion_Lasso_under_Information_ICCV_2023_paper.pdf.
[44] Z. Xie, L. Zhu, L. Zhao, B. Tao, L. Liu, W. Tao, Localization-aware channel pruning for object detection, Neurocomputing 403 (2020) 400–408, http://dx.doi.org/10.1016/j.neucom.2020.03.056.
[45] S. Li, L. Xue, L. Feng, Y. Wang, D. Wang, Object detection network pruning with multi-task information fusion, World Wide Web 25 (2022) 1667–1683, http://dx.doi.org/10.1007/s11280-021-00991-3.
[46] K. Simonyan, A. Vedaldi, A. Zisserman, Deep inside convolutional networks: Visualising image classification models and saliency maps, in: Workshop at International Conference on Learning Representations, ICLRW, 2014, http://dx.doi.org/10.48550/arXiv.1312.6034.
[47] Matthew D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: European Conference on Computer Vision, ECCV, Springer International Publishing, Cham, 2014, pp. 818–833, http://dx.doi.org/10.1007/978-3-319-10590-1_53.
[48] J.T. Springenberg, A. Dosovitskiy, T. Brox, M.A. Riedmiller, Striving for simplicity: The all convolutional net, in: Workshop at International Conference on Learning Representations, ICLRW, 2015, URL http://arxiv.org/abs/1412.6806.
[49] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, A. Torralba, Learning deep features for discriminative localization, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR, IEEE Computer Society, Los Alamitos, CA, USA, 2016, pp. 2921–2929, http://dx.doi.org/10.1109/CVPR.2016.319, URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2016.319.
[50] R.R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-cam: Visual explanations from deep networks via gradient-based localization, in: IEEE International Conference on Computer Vision, ICCV, 2017, pp. 618–626, http://dx.doi.org/10.1109/ICCV.2017.74.
[51] A. Chattopadhay, A. Sarkar, P. Howlader, V.N. Balasubramanian, Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks, in: IEEE Winter Conference on Applications of Computer Vision, WACV, 2018, pp. 839–847, http://dx.doi.org/10.1109/WACV.2018.00097.
[52] M. Jian, J. Wang, H. Yu, G. Wang, X. Meng, L. Yang, J. Dong, Y. Yin, Visual saliency detection by integrating spatial position prior of object with background cues, Expert Syst. Appl. 168 (2021) 114219, http://dx.doi.org/10.1016/j.eswa.2020.114219, URL https://www.sciencedirect.com/science/article/pii/S0957417420309441.
[53] J. Redmon, Darknet: Open source neural networks in c, 2013-2016, http://pjreddie.com/darknet/.
[54] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2016, pp. 770–778, http://dx.doi.org/10.1109/CVPR.2016.90.
[55] T.-Y. Lin, M. Maire, S.J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick, Microsoft coco: Common objects in context, in: European Conference on Computer Vision, ECCV, 2014, http://dx.doi.org/10.1007/978-3-319-10602-1_48.
[56] A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? the kitti vision benchmark suite, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2012, pp. 3354–3361, http://dx.doi.org/10.1109/CVPR.2012.6248074.
[57] Q. Lan, Q. Tian, Instance, scale, and teacher adaptive knowledge distillation for visual detection in autonomous driving, IEEE Trans. Intell. Veh. 8 (3) (2023) 2358–2370, http://dx.doi.org/10.1109/TIV.2022.3217261.
[58] J.I. Choi, Q. Tian, Adversarial attack and defense of yolo detectors in autonomous driving scenarios, in: IEEE Intelligent Vehicles Symposium, IV, 2022, pp. 1011–1017, http://dx.doi.org/10.1109/IV51971.2022.9827222.

Jung Im Choi is currently a Ph.D. student in Data Science at Bowling Green State University, Bowling Green, OH, USA. She received her MS degree in Computer Science from the University of Alabama in Huntsville, AL, USA, and her BE degree in Electronic and Electrical Engineering from Kyungpook National University, Daegu, Korea. Her research interests include the adversarial robustness of deep models, deep network pruning, and visual detection, especially in autonomous driving.

Qing Tian received a Ph.D. degree in electrical engineering in 2021 from McGill University, Montreal, QC, Canada. He is currently an assistant professor at the University of Alabama at Birmingham. From 2020 to 2023, he worked at Bowling Green State University, Bowling Green, OH, USA as an assistant professor. From 2019 to 2020, he was an applied scientist intern with Amazon.com, Inc. (Visual Search & AR) in Palo Alto, CA, USA. From 2013 to 2014, he worked as a software developer at Nakisa Inc., Montreal, QC, Canada. His research interests include autonomous driving visual perception, deep network compression, and adversarial AI. His current research in efficient and robust self-driving perception is supported by the National Science Foundation.