2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

BAM: Box Abstraction Monitors for Real-time OoD Detection in Object Detection

Changshun Wu[1], Weicheng He[1], Chih-Hong Cheng[2], Xiaowei Huang[3], and Saddek Bensalem[1]

***Abstract*— Out-of-distribution (OoD) detection techniques for deep neural networks (DNNs) become crucial thanks to their filtering of abnormal inputs, especially when DNNs are used in safety-critical applications and interact with an open and dynamic environment. Nevertheless, integrating OoD detection into state-of-the-art (SOTA) object detection DNNs poses significant challenges, partly due to the complexity introduced by the SOTA OoD construction methods, which require the modification of DNN architecture and the introduction of complex loss functions. This paper proposes a simple, yet surprisingly effective, method that requires neither retraining nor architectural change in object detection DNN, called Box Abstraction-based Monitors (BAM). The novelty of BAM stems from using a finite union of convex box abstractions to capture the learned features of objects for in-distribution (ID) data, and an important observation that features from OoD data are more likely to fall outside of these boxes. The union of convex regions within the feature space allows the formation of non-convex and interpretable decision boundaries, overcoming the limitations of VOS-like detectors without sacrificing real-time performance. Experiments integrating BAM into Faster R-CNN-based object detection DNNs demonstrate a considerably improved performance against SOTA OoD detection techniques, with a reduction in the false detection rate of over 10% in most cases.**

I. Introduction

Perception systems are critical for an autonomous robot and, among techniques to implement perception systems, object detection is a fundamental one. Despite remarkable advancements in performance, deep neural network (DNN) based object detection systems are not immune to safety concerns [1], [2], [3], [4], [5], particularly in critical applications that may affect human lives and crucial decision-making processes. For instance, as highlighted in [6], DNNs frequently exhibit noticeable performance deterioration when dealing with edge cases. Moreover, an additional mechanism is needed to reject, or assign low prediction confidence to, unknown samples, particularly those out-of-distribution (OoD) samples that are significantly different from the training dataset [7], [8], [9].

Nevertheless, integrating OoD detection into object detection DNNs while satisfying real-time requirements is known to be challenging. A recent experimental study [10] demonstrated that classical OoD detection algorithms such as softmax extensions [8], ODIN [11] or energy-based techniques [12], despite giving relatively satisfactory results in image classification tasks, do not yield satisfactory performance when directly applied to object detection. This pessimistic result leads researchers to consider altering of DNN architecture as well as retraining, e.g., VOS [10] and EvCenterNet [13]. Nevertheless, the training of such networks, as demonstrated by EvCenterNet, turns out to be non-trivial and hardly generalizable due to integrating multiple loss functions into one via introducing hyper-parameters. Worse still, in practice, the pre-trained models to be monitored may not be allowed to be modified.

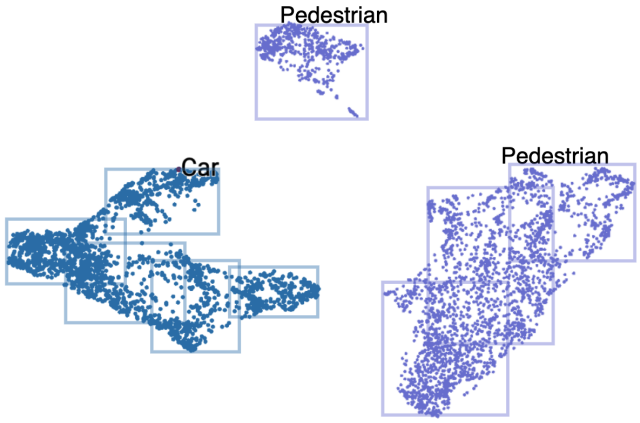

Fig. 1: An illustrative example demonstrating the superiority of BAM over the SOTA OoD detection method in object detection, VOS [10], which assumes a single center of the learned features of each output class and fits a class-conditional Gaussian distribution. However, a well-trained network does not necessarily form a single centered cluster for each output class (cf. the class of pedestrian). Even if it holds, the shape of the cluster does not necessarily have to be a n-dimensional ball (cf. the class of car). The data points in the figure represent the 2D projections of features generated by Faster R-CNN predictions (i.e, a bounding box and a classification label) at FC2 layer. These features were obtained using a Faster R-CNN model trained on the BDD100K dataset, and the dimensionality reduction was performed using UMAP [14].

In this paper, we show that such a route via architectural modification and retraining is not really mandatory, and propose Box Abstraction-based Monitor for Object Detection, BAM for short, a strikingly direct yet powerful method that empirically demonstrates superior performance compared to the SOTA, without any modification to a trained object

[1] Université Grenoble Alpes, Grenoble, France.
[2] Chalmers & University of Gothenburg, Gothenburg, Sweden.
[3] University of Liverpool, Liverpool, UK.
Correspondence to: changshun.wu@univ-grenoble-alpes.fr

detection DNN. BAM extends boxed abstraction monitors in classification [15], [16], [17], [18] to object detection. We propose leveraging the finite union of convex polytopes to enclose and characterize the shape of in-distribution (ID) data in the feature space. Our approach is rooted in a pragmatic observation, revealing that networks are not always able to learn a convex decision boundary with a unique center. This is exemplified in Fig. 1, where a natural way of enclosing the decision boundary for ID, as used by BAM, would be to use a union of multiple convex hulls. We also show that compared to existing results in boxed abstraction where data needs to be two-sided, for object detection, building the abstraction with one-sided data suffices to achieve considerable performance gain compared to existing results. Finally, we also detail additional architectural decisions, such as the layer where the monitor is introduced.

To meet stringent real-time requirements in object detection, we further consider the shape of the convex polyhedra to be used in the monitors and the number of polyhedra that can be introduced. With the parallelized computation introduced by the GPU, the benefit of using boxes allows highly efficient checking while the memory footprint for the boxes is substantially more compact than dedicated architecture pipelines such as EvCenterNet. Although our evaluation is done on Faster R-CNN [19] object detector with a two-stage architecture, we also detail how GPU parallelization enables migrating the techniques into single-stage detectors. Finally, we extensively evaluate BAM and compare its capabilities against sampling-free method VOS [10]. On all datasets including KITTI [20] and BDD100K [21] datasets, BAM demonstrates its superior performance for OoD detection with a reduction in the false detection rate of more than 10% in most cases, while only introducing 1.65% overhead compared to the standard Faster R-CNN implementation.

In summary, our contributions are as follows:

- A novel OoD detection framework integrated into two-stage object detectors without the need to modify or fine-tune the detection pipeline.
- An extensive evaluation against the SOTA baselines confirming the effectiveness of our methodology.
- A public repository on all our codes, models, and experimental results at `https://gricad-gitlab.univ-grenoble-alpes.fr/dnn-safety/bam-ood`.

The rest of the paper is organized as follows. After briefly reviewing related results in Sec. II, Sec. III presents details of our BAM approach. Experimental results are described in Sec. IV. Finally, the main conclusions and future work are drawn in Sec. V.

II. Related Work

Our proposed methodology is part of the active research field regarding OoD, novelty or anomaly detection, and uncertainty quantification (where one can set the proper uncertainty threshold for rejecting an input to be in-distribution), where the review from Salehi *et al.* [22] provides an excellent overview. However, one of the notable challenges is to migrate the technique into the object detection task while satisfying the real-time constraints by only using limited resources (memories to store the monitor and computing capabilities), which is the primary focus of this paper.

For OoD detection to be integrated into object detection, one natural method is to consider explicit or implicit ensembles, such as the MC dropout approach. This includes methods such as modifying the detection head to enable dropouts in SSD (Miller *et al.* [23]) or RetinaNet (Harakeh *et al.* [24]), as well as adding an additional dropout layer by extending YOLO (Kraus *et al.* [25]). Nevertheless, the ensemble method requires operating under multiple passes to generate the final decision regarding OoD, thus being computationally more expensive than sampling-free methods similar to our approach.

For sampling-free methods, we consider the closest work to ours to be VOS [10] due to the method also applicable on Faster R-CNN. While the empirical evaluation of our approach demonstrates superiority over VOS, the result can also be justified due to the theoretical analysis that VOS uses a convex shape for characterizing the ID data, while our method is more general than theirs due to the capability to have a finite union of convex polyhedra (boxes) to characterize the decision boundary for ID. The evidence can be observed by the generalizability VOS [10]; while it is considered as the SOTA when integrated into Faster R-CNN, its effectiveness diminishes in Transformer-based architectures [26]. The CertainNet [27] architecture extends the CenterNet [28] object detector by learning a set of class representatives called centroids, which are then compared with each prediction at inference time. When considering the decision boundary of centroids, it is again convex, thereby following similar limitations to VOS. Finally, EvCenterNet [13] also extends CenterNet by integrating evidential learning [29]. Both EvCenterNet and CertainNet require modifying the architecture while integrating new losses in the training process apart from the standard loss related to improving the prediction quality. Contrarily, our OoD detection approach can be directly integrated into any well-trained object detection network while ensuring real-time performance.

III. Technical Approach

We present our box abstraction-based monitor for Faster R-CNNs in two steps: i) introducing how to construct a box-based monitor for a Faster R-CNN (Sec. III-B); ii) explaining how to utilize it with the network to identify potential mispredictions in real-time (Sec. III-C).

A. Basic Notions

Before introducing the boxed monitor, we review the essential concepts and notation relating to Faster R-CNN networks and the key element in our method, *box abstractions*, an efficient data structure for representing the features corresponding to network detections from ID data. Let $\mathbb{N}$ and $\mathbb{R}$ be the sets of natural and real numbers. We use $[a \cdots b]$ with $a, b \in \mathbb{N}$ and $a \leq b$ to refer to integer intervals. To refer

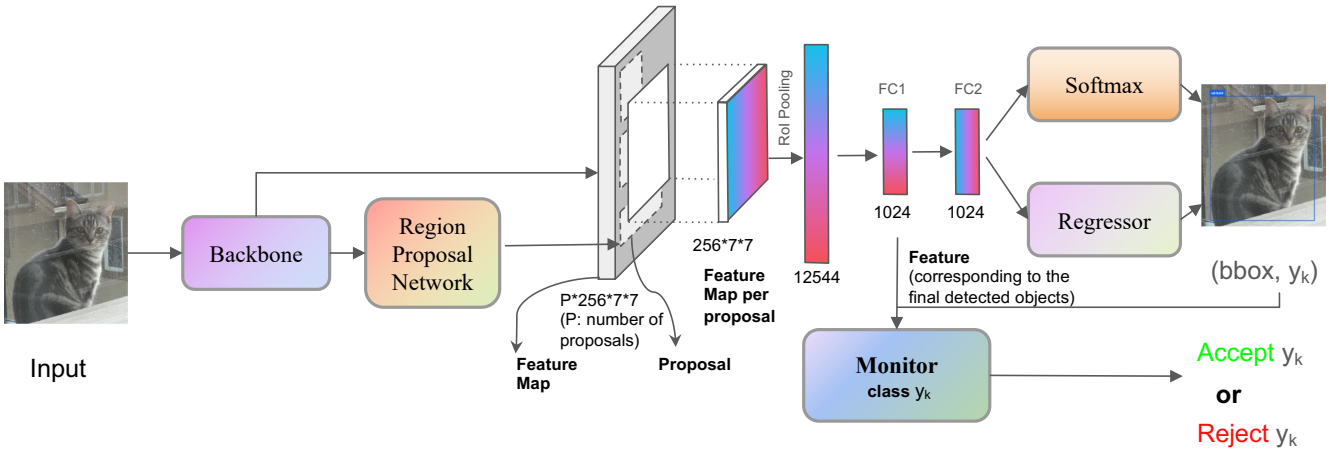

Fig. 2: Faster R-CNN architecture and the integration of BAM. For monitor construction, features are extracted from FC1 or the penultimate layer FC2 in the MLP Head of the model. The value P, i.e., the number of proposals per image, equals 1000.

to real intervals, we use $[a, b]$ with $a, b \in \mathbb{R} \cup \{-\infty, \infty\}$ and if $a, b \in \mathbb{R}$, then $a \leq b$. We use a square bracket when both sides are included and a round bracket to exclude endpoints (e.g., $[a, b)$ for excluding b). For $n \in \mathbb{N} \setminus \{0\}$, $\mathbb{R}^n \stackrel{\text{def}}{=} \underbrace{\mathbb{R} \times \cdots \times \mathbb{R}}_{n \text{ times}}$ is the space of real coordinates of dimension n and its elements are called n-dimensional vectors. We use $\mathbf{x} = (x_1, \ldots, x_n)$ to denote an n-dimensional vector and $\mathbf{x}_j$ as the j-th element x_j for $j \in [1 \cdots n]$.

1) Faster R-CNN models: A Faster R-CNN model typically consists of three key components: a backbone network, a region proposal network (RPN), and a region of interest (RoI) head, as shown in Figure 2. In general, the model operates according to the following workflow. The backbone network initially acquires an input image and generates a corresponding feature map. Subsequently, the RPN utilizes this feature map and anchor information to generate a set of object proposals, each consisting of a tuple of tentative bounding box coordinates and an objectness score indicating the likelihood of containing an object of interest. Afterward, the RoI head proceeds to process these proposals further, extracting a fixed-length feature vector for each proposal. Finally, these feature vectors are fed into the Multi-layer perceptron (MLP) head (classifier) for predicting the object category and determining the ultimate bounding box position.

2) Tight box abstraction for a dataset [15], [16], [18]: In n-dimensional geometric space, a *box* is a continuous set, often used to abstract some point sets. It consists of n intervals, each corresponding to the upper and lower bounds that each dimension can take. For a dataset $X = \{\mathbf{x}^1, \ldots, \mathbf{x}^m\}$, we define its *tight box abstraction* as $\mathsf{TBA}(X) \stackrel{\text{def}}{:=} \langle [a_1, b_1], \ldots, [a_n, b_n] \rangle$ as an n dimensional box, where for $i \in [1 \cdots n]$, $a_i = \mathsf{min}(\{\mathbf{x}_i^j\})$ and $b_i = \mathsf{max}(\{\mathbf{x}_i^j\})$.

3) Distance between a data point and a box abstraction: Given a data point $\mathbf{x} = (x_1, \ldots, x_n)$ and a box $B = \langle [a_1, b_1], \ldots, [a_n, b_n] \rangle$, the distance between them, denoted as $d(\mathbf{x}, B)$, is defined as the sum over the distances between each x_i and the interval $[a_i, b_i]$, which is defined as follows:

$$d(x_i, [a_i, b_i]]) = \begin{cases} 0 & \text{if} \quad a_i \leq x_i \leq b_i \\ a_i - x_i & \text{if} \quad x_i < a_i \\ x_i - b_i & \text{if} \quad x_i > b_i \end{cases} \tag{1}$$

Intuitively, this distance measures the effort required to move an external data point into a box, under the assumption that each dimension is independent.

4) Enlargement of box abstraction: A tight box for a dataset can be easily enlarged to encompass by using a buffer vector $\delta \stackrel{\text{def}}{:=} [\delta_1, \ldots, \delta_n]$ to relax the lower and upper bounds as follows: $B_\delta = \langle [a_1 - \delta_1, b_1 + \delta_1], \ldots, [a_n - \delta_n, b_n + \delta_n] \rangle$. For example, consider a box $B_1 = \langle [0.1, 0.3], [0.2, 0.5] \rangle$ and an outside point $\mathbf{x} = (0.5, 0.6)$. To include this point within the abstraction, the box can be easily enlarged as follows: $B_1' = \langle [0.1, 0.3 + (0.5 - 0.3)], [0.2, 0.5 + (0.6 - 0.5)] \rangle = \langle [0.1, 0.5], [0.2, 0.6] \rangle$.

B. Monitor Construction

1) Box-based monitor construction for Faster R-CNNs: For an object detection network with fixed parameters, let $D_{\text{train}} \stackrel{\text{def}}{:=} \{(\mathbf{x}, \mathbf{gt})\}$ represent its training dataset *with* $\mathbf{x}$ being the input and $\mathbf{gt}$ being the associated ground-truth labels. For an input $\mathbf{x}$, let $prop_i \in f_{bbrpn}(\mathbf{x})$ be one of the region proposals generated by the backbone and RPN network of the Faster R-CNN, and let $p_i \stackrel{\text{def}}{:=} (bbox_i, cls_i) = f_{mlp}(prop_i)$ be the output predictions for proposal $prop_i$ passing through the MLP, where $bbox_i$ is the predicted bounding box in terms of size, location and orientation, and cls_i is the *output class* of the predicted bounding box. Let $f_{mlp}^l(prop_i)$ be the feature vector at the l-th layer where the monitor is constructed. Based on the previous usage of box abstraction [15], [18], boxes are generally constructed at fully-connected layers closer to the output, as they provide a more high-level representation of features of input data.

Assume that the Faster R-CNN can produce Y output classes indexed from 1 to Y. We now describe how to construct a box abstraction monitor $\mathcal{B}_y^l = \{B_y^{l,1}, \ldots, B_y^{l,k_y}\}$ at layer l for each class $y \in [1 \cdots Y]$ of a Faster R-CNN network using a four-step process shown in Figure 3:

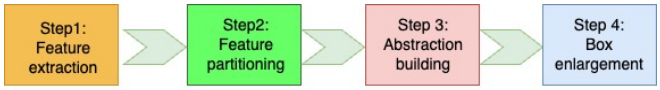

Fig. 3: Process Flow Diagram for Constructing Box Abstraction Monitors.

- **(Step 1: Feature extraction)** For each output class $y \in [1, \ldots, Y]$, Construct F_y^l to contain all feature vectors at the l-th layer that contribute to a bounding box prediction [1] of class y, as formulated in Eq. (2).

$$F_y^l \stackrel{\text{def}}{:=} \{f_{mlp}^l(prop_i) \mid prop_i \in f_{bbrpn}(\mathbf{x}), (\mathbf{x}, \mathbf{gt}) \in D_{\text{train}}, (bbox_i, y) = f_{mlp}(prop_i)\} \quad (2)$$

  Once the feature extraction step is completed, each feature vector is currently implemented in BAM as a one-dimensional tensor, containing 1024 neuron activation values for a detected object.
- **(Step 2: Feature vector partitioning)** The set of feature vectors in F_y^l is partitioned into k_y subsets, denoted as $\pi(F_y^l) \stackrel{\text{def}}{:=} \{F_y^{l,1}, \ldots, F_y^{l,k_y}\}$. The partitioning is done by applying a *k-means* [30] algorithm to cluster these features: the number of clusters is decided by the *density*, which is the targeted number of data points within each cluster. Given a dataset of size m and the envisioned density value ρ, the number of clusters to be used in the k-means algorithm equals $\lfloor \frac{m}{\rho} \rfloor$. The density is thus a hyper-parameter that decides the number of resulting boxes for a given dataset. To avoid cases when the dataset turns huge, we constraint the maximum k from above with a constant T, reflecting the real-time processing capability of the underlying hardware. In our evaluation, we configure T to be 10000, while k never exceeds 8000.
- **(Step 3: Abstraction building)** Construct $\mathcal{B}_y^l \stackrel{\text{def}}{:=} \{B_y^{l,1}, \ldots, B_y^{l,k_y}\}$, where for $j \in [1 \cdots k_y]$, $B_y^{l,j}$ is a tight box abstraction for subset $F_y^{l,j}$, i.e., $B_y^{l,j} = \mathsf{TBA}(F_y^{l,j})$.
- **(Step 4: Box enlargement)** The enlargement of boxes is driven by the need to control the true positive rate (TPR) within the in-distribution dataset. To ensure a fair comparison with VOS which uses FPR95 (i.e., false positive rate of OoD samples at a 95% true positive rate of ID samples) as the evaluation criterion, we also enlarge the box based on FPR95. This is done algorithmically by first sorting all feature vectors (with distance from small to large) that fall outside the boxes created by Step 3. Subsequently, enlarge the box by including sorted feature vectors until TPR95 is reached. Given a feature vector $\mathbf{z}$, its distance to the monitor $\mathcal{B}_y^l$ is defined using Eq. (3), which is the minimum distance to any box in the monitor.

$$\mathsf{dist}(\mathbf{z}, \mathcal{B}_y^l) \stackrel{\text{def}}{:=} \min_{j \in [1 \cdots k_y]} \{d(\mathbf{z}, B_y^{l,j})\} \quad (3)$$

[1] Similar to VOS, we set the confidence score threshold for predictions to maximize the Faster R-CNN model's micro F1 score, thereby preventing low-scoring predictions from being considered as relevant features for monitor construction.

C. Monitor Deployment

Given a neural network N and the boxed abstraction monitor $\{\mathcal{B}_1^l, \ldots, \mathcal{B}_Y^l\}$, in runtime, the **monitor rejects a class y prediction $p_i \stackrel{\text{def}}{:=} (bbox, y)$ for an input $\mathbf{x}$** generating region proposal $prop_i$, if $\nexists j \in [1 \cdots k_y] : f^l(prop_i) \in B_y^{l,j}$. That is, no box contains the feature vector $f^l(prop_i)$ produced at the l-th layer. As the containment checking $f^l(prop_i) \in B_y^{l,j}$ compares $f^l(prop_i)$ against the box's lower and upper bounds on each dimension, it can be done in time *linear to the number of monitored neurons*.

In summary, the core idea of BAM is to use a data structure to properly enclose the regions in the feature space where the neural network has made decisions. When the DNN makes a prediction, BAM evaluates whether the corresponding features for making the prediction fall within the enclosed regions. If the answer is positive, this decision can be considered similar to some previously observed decision behaviors, which are deemed ID. Otherwise, the decision is diagnosed by BAM as OoD.

Remark We chose to use a box instead of other geometric shapes or probability distributions for two main reasons. First, the box allows for the most efficient membership query, requiring only straightforward comparisons of real numbers across n dimensions. In contrast, other shapes require more complex calculations, such as linear summations or multiplications. Second, we opted against a distribution fitting approach due to the uneven data distribution within each cluster. When a cluster has very few data points, distribution fitting can become highly inaccurate. While our methodology is illustrated in two-stage detectors, we briefly summarize how to transfer it to the single-stage detector. Single-stage detectors such as YOLO or SSD maintain a grid of cells, with each cell responsible for predicting if there is an object that is centered in that cell. A simple (brute-force) method is to create, for each cell, one box abstraction-based monitor.

IV. Implementation and Experiments

In this section, we present the implementation of BAM and experimental results that validate its effectiveness on multiple real-world object detection datasets and network variants.

A. Implementation

We have implemented BAM using *PyTorch* [31], *Scikit-learn* [32], the computer vision library *Detectron2* [33], and the object detection dataset management library *Fifty-one* [34], where in the implementation, we have developed three utility modules for *feature extraction*, *clustering*,

and *abstraction building*. Specifically, the feature extraction module relies on the functionality provided by Detectron2, enabling the loading of the Faster R-CNN model and the generation of predictions on a given dataset. By default, the Faster R-CNN model can only output the bounding box coordinates for object localization and the classification confidence score. To facilitate the feature extraction from the intermediate layers of the model, we have implemented a custom forward function extended from Faster R-CNN architecture within Detectron2.

B. Experiment Setup

We tested on diverse ID datasets to validate the presented approach, utilizing Faster R-CNN models with different backbones. To assess the monitor performance of each model variant, we evaluated them against multiple OoD datasets. In the following, we describe the used datasets, models, the evaluation metric, and the process in detail.

1) Datasets: We trained our model using two ID object detection datasets specific to the autonomous driving domain: BDD100K [21] and KITTI [20]. To evaluate the effectiveness of our monitor in detecting OoD data, we evaluated it on three OoD datasets. Among these, we use two datasets previously employed in VOS: MS-COCO [35] and OpenImages [36]. Additionally, we curated an extra OoD dataset derived from PASCAL VOC [37], in which we manually filtered out images containing ID objects.

2) Models: In this study, our primary focus is the Faster R-CNN object detection architecture. We trained Faster R-CNN models with a feature pyramid network (FPN) using three backbone architectures: ResNet-50, ResNet-101, and RegNetX-4.0GF. To attain satisfactory performance, each model variant is fine-tuned for a minimum of 40 epochs, starting from the pre-trained weights on the ImageNet dataset.

3) Metrics: In OoD detection, a "true positive" refers to an object identified as an ID object that does indeed fall within the ID categories. In contrast, a "false positive" means an object identified as an ID object that actually belongs to the OoD categories. We assess the monitor performance in OoD detection using the standard metric *FPR95*, which represents the false positive rate of OoD samples at a 95% true positive rate of ID samples. A lower FPR95 indicates a superior ability of monitor to detect OoD objects while still encompassing 95% of the true positives from the ID datasets.

4) Process: To evaluate the developed monitors, performance tests are conducted using OoD datasets. This assessment allows us to gauge the monitors' capability to accurately identify and reject OoD objects during testing. At the end of the evaluation process, a confusion matrix is constructed for the monitors, facilitating the computation of FPR95. These evaluation results are then compared with VOS, which is regarded as the SOTA of OoD detection methods in object detection.

C. Results

In this section, we present experimental results to demonstrate the effectiveness of our method BAM compared to VOS, which is commonly regarded as the SOTA against classical methods such as Softmax [8], ODIN [11], Energy [12].

TABLE I: Comparing BAM (feature vector from FC2Relu) with the state-of-the-art VOS method; ↑ indicates larger values are better and ↓ indicates smaller values are better; numbers in **bold** texts imply superiority of the method, with at least 10% performance increase. Values being underlined imply the performance of two methods being on par.

ID	Backbone	mAP↑ (ID)	Method	FPR95 ↓ (OoD: MS-COCO/OpenImage/VOC)
BDD100k	ResNet50	31.5	VOS	48.93 / 41.85 / 53.56
			BAM	**37.77** / **31.25** / **47.58**
	ResNet101	32.5	VOS	37.27 / 21.26 / 45.70
			BAM	**23.41** / **8.52** / **32.47**
	RegX4.0	32.7	VOS	42.91 / 34.16 / 46.77
			BAM	38.87 / **26.74** / 47.7
KITTI	ResNet50	79.5	VOS	15.97 / 7.51 /19.03
			BAM	**3.69** / **1.63** / **5.24**
	ResNet101	86.2	VOS	6.09 / 2.37 / 14.52
			BAM	**4.24** / 2.54 / **8.98**
	RegX4.0	79.2	VOS	12.52 / 5.46 / 14.87
			BAM	**7.48** / 5.07 / **9.88**

1) Effectiveness: Our BAM approach achieves superior performance in terms of FPR95 across datasets. In Table I, we summarize the comparison between BAM and VOS. In many cases, 14 out of 18, BAM outperforms VOS by a significantly large margin ($> 10\%$), without any case being under-performed drastically. Even when we refine the margin to be 5%, out of 18 OoD cases, BAM is still superior to VOS in 16 datasets and on par in 1 datasets, with 1 being worse. Finally, when comparing performance against absolute values, BAM remains superior to VOS in 16 cases.

Fig. 4 presents a qualitative analysis of predictions on several OoD images using object detection models with the benchmark method VOS (top) and BAM (bottom). We use KITTI as the ID dataset. As illustrated by the green bounding boxes in the figure, BAM outperforms the VOS OoD detector in terms of identifying OoD objects, and reducing false positives among detected objects.

2) Execution Time: On Nvidia RTX A4000 8GB, the GPU-based inference time with BAM averages 41.1 milliseconds per detection instance on the KITTI dataset, compared to 40.8 milliseconds without monitoring. This signifies that the monitoring module introduces a negligible additional overhead of 0.7% (0.3 milliseconds). On the BDD dataset, specifically in the case of the monitor consisting of 7000 boxes for the "car" category, the inference time with BAM increases to an average of 94.4 milliseconds for a single image containing 1000 region proposals, as opposed to the original time of 85.4 milliseconds without monitoring. Therefore, our BAM approach enhances the inference process without significantly impacting the real-time performance.

3) Ablation study (the impact of layer selection): We conducted further investigations to explore the impact of utilizing features from different fully connected layers for

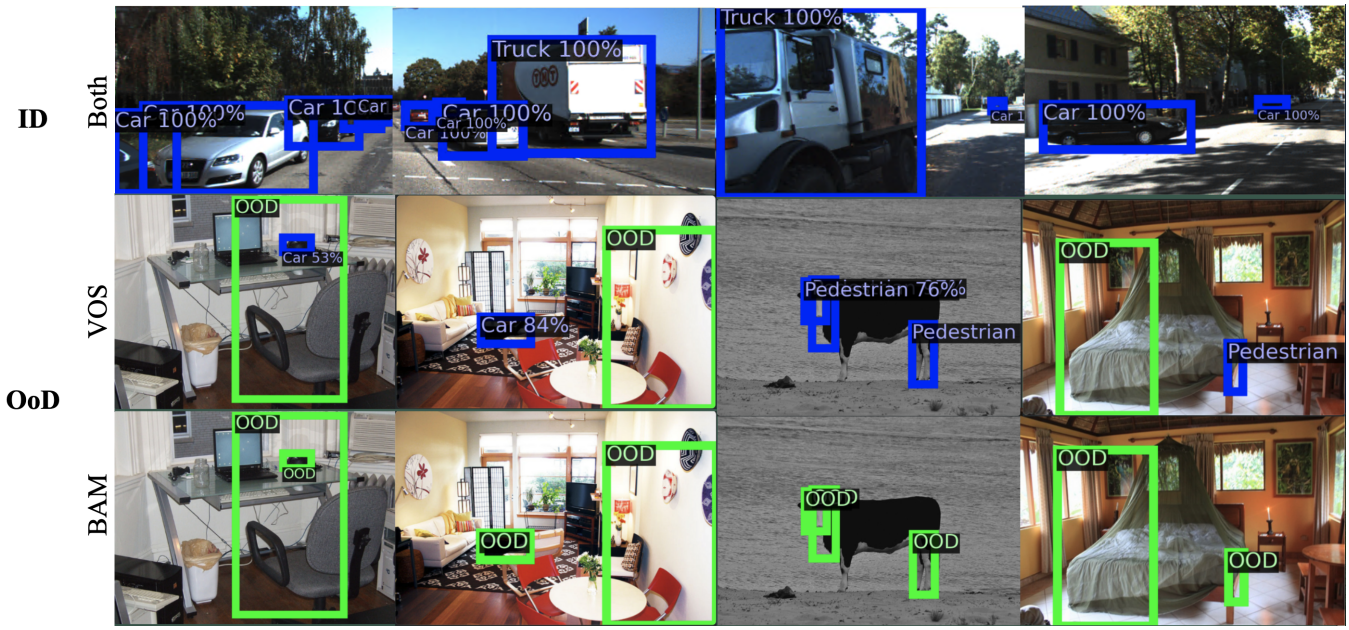

Fig. 4: Visualization of OoD detection results using the VOS and BAM methods: The first row shows ID images from the KITTI dataset, demonstrating that both methods accurately classify ID objects. The second and third rows display OoD images from the MS-COCO dataset, processed by VOS and BAM, respectively, highlighting BAM's superior ability to reduce false positives among detected objects. Blue: Objects classified as ID. Green: OoD objects detected by VOS or BAM.

monitor construction. Specifically, we evaluated the performance of the monitors built at layer FC2 and the preceding layer FC1, both with the ReLU activation function. The representative results are shown in Table II. Overall, we can observe that the selection of layers can create fluctuation in the performance, as for the KITTI dataset, selecting FC1Relu can achieve better performance. Nevertheless, as demonstrated in Table I, even when selecting the least-performing FC2Relu, the performance is still strictly better than that of VOS.

TABLE II: Performance comparison of BAM monitors with features extracted from different layers in the MLP Head. The monitors' performances are consistent across different layers.

ID	Backbone	Layer	FPR95 (OoD: MS-COCO/OpenImage/VOC-OoD)
KITTI	ResNet50	FC1Relu	3.77 / 3.35 / 6.05
		FC2Relu	**3.69** / **1.63** / **5.24**
	RegX4.0	FC1Relu	**7.23** / 7.25 / **9.01**
		FC2Relu	7.48 / **5.07** / 9.88

4) Ablation study (the impact of cluster density): In this ablation study, we vary the hyper-parameter density ρ to assess its impact on the performance. Remarkably, the results revealed a notable reduction in FPR95 across all density settings, as illustrated in Fig. 5. While the study shows that the performance is not sensitive to ρ, the significant decrease in FPR95 across all ρ values compared to VOS underscores the true effectiveness of the proposed approach. Furthermore, one should note that the density of data points should not be minimized excessively, as we are using a finite set of data points to approximate an infinite set. If the density is too low, it can lead to issues similar to overfitting in deep learning.

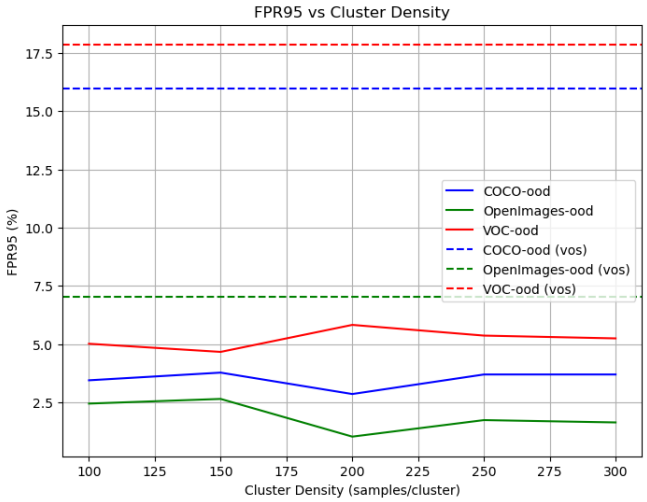

Fig. 5: Ablation study on the hyper-parameter ρ, density of data points within each cluster. In all settings (varying ρ on x-axis from 100 to 300), our method BAM is better than VOS and performs consistently.

V. Conclusion

This paper presented BAM, a box-abstraction-based OoD monitoring method for object detection. BAM nicely enables

the characterization of complex and non-convex OoD decision boundaries in the feature space using a finite union of boxes. In addition, BAM can be introduced without the need to change the standard object detection network while maintaining real-time detection capabilities. Our experimental results outperformed the state-of-the-art method by achieving a lower false positive rate of OoD samples while reaching a true positive rate of 95% for ID samples. Regarding future research directions, we aim to implement this method in other object detection model families such as YOLO and CenterNets by deciphering the feature representations of these specific learning architectures. The second direction is to develop a principled approach regarding how to perform refinement when the abstraction is too coarse, as coarse abstraction can negatively influence the decision boundary to manifest false alarms. A potential solution is to explore combining geometric shape abstractions with the density distribution of features within those shapes. Yet another direction is to consider refining the algorithm by taking into account that a region proposal can have multiple objects. Finally, we aim to consider how the construction of monitors can be aligned with safety principles with a clearly specified data quality requirement, such as including a database of edge cases or rare events occurring on the road. This enables an objective evaluation method that is not biased towards specific datasets.

REFERENCES

[1] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, and X. Yi, "A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability," *Computer Science Review*, vol. 37, p. 100270, 2020.

[2] D. Hendrycks, S. Basart, N. Mu, S. Kadavath, F. Wang, E. Dorundo, R. Desai, T. Zhu, S. Parajuli, M. Guo, *et al.*, "The many faces of robustness: A critical analysis of out-of-distribution generalization," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 8340–8349, 2021.

[3] S. Abrecht, A. Hirsch, S. Raafatnia, and M. Woehrle, "Deep learning safety concerns in automated driving perception," *arXiv preprint arXiv:2309.03774*, 2023.

[4] R. Dwivedi, D. Dave, H. Naik, S. Singhal, R. Omer, P. Patel, B. Qian, Z. Wen, T. Shah, G. Morgan, *et al.*, "Explainable ai (xai): Core ideas, techniques, and solutions," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–33, 2023.

[5] B. Li, P. Qi, B. Liu, S. Di, J. Liu, J. Pei, J. Yi, and B. Zhou, "Trustworthy ai: From principles to practices," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–46, 2023.

[6] K. Li, K. Chen, H. Wang, L. Hong, C. Ye, J. Han, Y. Chen, W. Zhang, C. Xu, D.-Y. Yeung, *et al.*, "CODA: A real-world road corner case dataset for object detection in autonomous driving," in *ECCV*, pp. 406–423, Springer, 2022.

[7] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 427–436, 2015.

[8] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," in *ICLR*, 2016.

[9] J. Yang, K. Zhou, Y. Li, and Z. Liu, "Generalized out-of-distribution detection: A survey," *International Journal of Computer Vision*, pp. 1–28, 2024.

[10] X. Du, Z. Wang, M. Cai, and Y. Li, "VOS: Learning what you don't know by virtual outlier synthesis," *ICLR*, 2022.

[11] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," in *ICLR*, 2018.

[12] W. Liu, X. Wang, J. Owens, and Y. Li, "Energy-based out-of-distribution detection," *NeurIPS*, vol. 33, pp. 21464–21475, 2020.

[13] M. R. Nallapareddy, K. Sirohi, P. L. Drews-Jr, W. Burgard, C.-H. Cheng, and A. Valada, "EvCenterNet: Uncertainty estimation for object detection using evidential learning," in *IROS*, pp. 5699–5706, IEEE, 2023.

[14] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[15] T. A. Henzinger, A. Lukina, and C. Schilling, "Outside the box: Abstraction-based monitoring of neural networks," in *ECAI*, pp. 2433–2440, IOS Press, 2020.

[16] C.-H. Cheng, C.-H. Huang, T. Brunner, and V. Hashemi, "Towards safety verification of direct perception neural networks," in *DATE*, pp. 1640–1643, IEEE, 2020.

[17] C.-H. Cheng, C. Wu, E. Seferis, and S. Bensalem, "Prioritizing corners in ood detectors via symbolic string manipulation," in *ATVA*, pp. 397–413, Springer, 2022.

[18] C. Wu, Y. Falcone, and S. Bensalem, "Customizable reference runtime monitoring of neural networks using resolution boxes," in *RV*, pp. 23–41, Springer, 2023.

[19] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *NeurIPS*, vol. 28, 2015.

[20] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *CVPR*, pp. 3354–3361, IEEE, 2012.

[21] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "BDD100K: A diverse driving dataset for heterogeneous multitask learning," in *CVPR*, pp. 2636–2645, IEEE, 2020.

[22] M. Salehi, H. Mirzaei, D. Hendrycks, Y. Li, M. Rohban, M. Sabokrou, *et al.*, "A unified survey on anomaly, novelty, open-set, and out of-distribution detection: Solutions and future challenges," *Transactions on Machine Learning Research*, no. 234, 2022.

[23] D. Miller, L. Nicholson, F. Dayoub, and N. Sünderhauf, "Dropout sampling for robust object detection in open-set conditions," in *ICRA*, pp. 3243–3249, IEEE, 2018.

[24] A. Harakeh, M. Smart, and S. L. Waslander, "BayesOD: A Bayesian approach for uncertainty estimation in deep object detectors," in *ICRA*, pp. 87–93, IEEE, 2020.

[25] F. Kraus and K. Dietmayer, "Uncertainty estimation in one-stage object detection," in *ITSC*, pp. 53–60, IEEE, 2019.

[26] X. Du, G. Gozum, Y. Ming, and Y. Li, "Siren: Shaping representations for detecting out-of-distribution objects," *NeurIPS*, vol. 35, pp. 20434–20449, 2022.

[27] S. Gasperini, J. Haug, M.-A. N. Mahani, A. Marcos-Ramiro, N. Navab, B. Busam, and F. Tombari, "CertainNet: Sampling-free uncertainty estimation for object detection," *RA-L*, vol. 7, no. 2, pp. 698–705, 2021.

[28] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "CenterNet: Keypoint triplets for object detection," in *ICCV*, pp. 6569–6578, IEEE, 2019.

[29] M. Sensoy, L. Kaplan, and M. Kandemir, "Evidential deep learning to quantify classification uncertainty," *NeurIPS*, vol. 31, pp. 3183–3193, 2018.

[30] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[31] A. Paszke, , and et al., "Pytorch: An imperative style, high-performance deep learning library," *NeurIPS*, vol. 32, pp. 8024–8035, 2019.

[32] F. Pedregosa and et al., "Scikit-learn: Machine learning in Python," *Journal of ML Research*, vol. 12, pp. 2825–2830, 2011.

[33] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2." `https://github.com/facebookresearch/detectron2`, 2019.

[34] B. E. Moore and J. J. Corso, "Fiftyone," *GitHub Note: https://github.com/voxel51/fiftyone*, 2020.

[35] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *ECCV*, pp. 740–755, Springer, 2014.

[36] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, T. Duerig, and V. Ferrari, "The open images dataset v4," *IJCV*, vol. 128, no. 7, pp. 1956–1981, 2020.

[37] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *IJCV*, vol. 88, no. 2, pp. 303–338, 2010.