Enabling Effective OOD Detection via Plug-and-Play Network for Mobile Visual Applications

Zixiao Wang, Qi Dong, Tianzhang Xing, Zhidan Liu, Zhenjiang Li, Xiaojiang Chen

Abstract—Mobile devices have increasingly integrated with numerous deep learning-based visual applications, such as object classification and recognition models. While these models perform well in controlled environments, their effectiveness declines in real-world environment due to out-of-distribution (OOD) data not seen during training. Existing methods for detecting OOD data often compromise normal data recognition and require extensive training on unattainable OOD data. To address these issues, we propose POD, a framework designed to enhance mobile visual applications by providing high-precision OOD detection without affecting original model performance. In the offline phase, POD generates OOD detectors from any classification model by analyzing model's neuron responses to various data types. In the online phase, it continuously adjusts decision boundaries by integrating results from both the original model and the detector. Evaluated on two public datasets and one self-collected dataset across various popular classification models, POD significantly improves OOD detection performance while maintaining the accuracy of original models.

Index Terms—Mobile visual applications, OOD detection, decision boundary

1 Introduction

With the rapid advancement of deep learning technology, deep learning models have become increasingly prevalent in a variety of well-defined mobile visual applications, such as facial recognition for payments and access control [1]–[3]. Among these applications, classification and recognition models are particularly popular and show promise for use in more diverse and challenging environments, such as autonomous driving [4] and drug detection [5]. Despite this potential, deep learning models currently play a supporting role in assisting human decision-making in these applications. In comparison to controlled environments, the reliability of deep learning model results in real-world environment is compromised, necessitating additional human verification or intervention [6]. This is primarily because classification and recognition models may encounter objects that belong to unfamiliar categories, known as *out-of-distribution* (OOD) data, leading them to misclassify these unseen objects as the most similar category encountered during training, known as *in-distribution* (ID) data.

Therefore, a critical technical challenge in enabling the widespread application of classification models in real-world scenarios is the identification and filtering of OOD data. This process is essential for transforming the classification task into a familiar and deterministic environment for the model. Various research efforts have been dedicated to addressing this challenge. Methods based on maximum softmax probability and its variants [7], [8] are simple to implement but often suffer from overconfidence and a lack of deep feature representation. Generative model-based approaches [9], [10] tend to encounter issues such as training instability, model fitting bias, and confusion between the likelihoods of OOD and ID samples, especially in high-dimensional data scenarios. Distance-based methods [11], [12] rely heavily on high-quality feature extraction and accurate statistical estimation, making them sensitive to input noise and lacking robustness. A popular decision-boundary-based OOD detection strategy [13], [14] synthesizes virtual OOD samples near the classifier's decision boundaries to optimize the model's performance in regions of uncertainty, and has demonstrated promising detection performance. Although such a strategy partially addresses the limitations mentioned above, its effectiveness is still constrained by the limited quality of the synthesized samples, making it difficult to robustly handle the complex and diverse OOD scenarios encountered in real-world environments.

Drawing inspiration from the concept of plug-and-play design, which enhances model performance by introducing separate and lightweight modules [15], [16], we propose the integration of an OOD detector module. This module is designed to assist a classification model in identifying OOD data without compromising its accuracy on ID data. The OOD detector first assesses the input data to determine if it can be accurately classified by the model. If the data is identified as ID data, the classification model proceeds with generating a reliable result; otherwise, the data is labeled as OOD and skipped by the classification model. However, developing an effective OOD detector poses significant challenges, primarily due to the following two reasons.

Firstly, numerous classification models have been devel-

- *Zixiao Wang, Qi Dong, Tianzhang Xing, and Xiaojiang Chen are with Northwest University, China. Tianzhang Xing and Xiaojiang Chen are also with the Shaanxi Key Laboratory of Passive Internet of Things and Neural Computing, China. E-mails: wangzixiao@stumail.nwu.edu.cn, dongdidong42@gmail.com, {xtz, xjchen}@nwu.edu.cn.*
- *Zhidan Liu is with the Hong Kong University of Science and Technology (Guangzhou), China. E-mail: zhidanliu@hkust-gz.edu.cn.*
- *Zhenjiang Li is with the City University of Hong Kong, China. E-mail: zhenjiang.li@cityu.edu.hk.*
- *Corresponding authors: Tianzhang Xing and Zhidan Liu.*

oped, offering a range of options to select the most suitable model tailored to the requirements of a specific classification task for optimal performance. However, due to the diverse designs of these models, there is a lack of comprehensive studies on designing effective OOD detectors that can seamlessly integrate with different classification models.

Secondly, acquiring a sufficient amount of data to train an OOD detector remains a challenge, even after identifying the optimal architectural design. Additionally, once deployed in real-world environment, ongoing adjustments to the OOD detector are essential to address any disparities between the training dataset and real-world data.

To tackle these challenges, we introduce POD, a plug-and-play network-based OOD detection approach. This innovative method automatically generates OOD detectors for any classification model, enabling them to complement the original classifier. The derived OOD detector identifies OOD data and performs predefined actions, such as filtering or alerting the user. Our key insight lies in leveraging a well-trained classifier's proficiency in processing ID data. By inputting an initial OOD dataset into the classifier, we can identify the crucial model components for OOD detection by analyzing and contrasting the internal neural responses to both ID and OOD data. These OOD-sensitive components then form an OOD detector specific to the classifier. This approach allows us to derive an OOD detector directly from the original classification model, which we found to be versatile and applicable to various mainstream models. In scenarios where memory-constrained edge devices require OOD detectors, POD offers efficient model compression to meet memory constraints with minimal performance impact. Furthermore, POD provides an auxiliary detector for situations where the original classifier is a black box without accessible components.

Moreover, during the offline training phase, POD integrates an OOD data augmentation technique to generate OOD data by transforming existing ID data. This approach allows the OOD detector to be trained using augmented data, eliminating the need to collect additional OOD datasets. In real-world applications, POD can improve the detector's performance by leveraging output from both the original classifier and the OOD detector to dynamically update the detector model.

The innovative design of POD not only enables service providers to seamlessly integrate OOD detection functionality into their classification models but also functions as a third-party plugin to equip models or devices with OOD detection capabilities. We have developed a prototype of POD and utilized various specialized and general models to create their respective OOD detectors. Performance evaluations have been conducted using two real-world datasets on different platforms, including an NVIDIA GeForce RTX 4070, Raspberry Pi 4B, and Xiaomi smartphone. Extensive experiments demonstrate that POD surpasses state-of-the-art methods, achieving an average FPR95 reduction of 31.24% and an AUROC improvement of 11.50%. Additionally, when utilizing the lightweight general auxiliary detector, POD introduces a maximum latency of only 137 ms during online inference. Furthermore, we observe a continuous enhancement in POD's performance with an increase in available samples during the online phase.

The key contributions of this study are as follows:

- We introduce POD to tackle the challenge of OOD detection for popular models on mobile devices in real-world environment, facilitating the identification of OOD data without compromising the models' classification performance.
- We develop a lightweight OOD detector generation method by constructing a compact detector based on the backbone network of the original model. Furthermore, we define two distinct types of OOD data to better align with the proposed framework and enhance detection performance.
- We introduce an adaptive decision boundary update technique that dynamically adjusts decision criteria during the inference phase, continuously optimizing OOD detection capabilities to accommodate diverse unknown samples.
- We implemented a prototype of POD and conducted extensive experiments that showcase significant performance improvements compared to state-of-the-art OOD detection methods.

The rest of the paper is organized as follows: Section 2 provides background on OOD detection. Section 3 introduces the overall architecture of the proposed POD framework. Sections 4 and 5 present the design details of POD. Section 6 describes the experimental setup. In Section 7, we implement POD and evaluate its performance. Section 8 reviews related work, and Section 9 concludes the paper.

2 Background

2.1 Classification Models in the Wild

In the realm of machine learning and deep learning models, particularly in object classification tasks, it is typically assumed that training data is drawn from a specific distribution. However, in real-world scenarios, these models may confront data from distributions that deviate from the training data distribution, a phenomenon known as *out-of-distribution* (OOD) data. Classifying OOD data based on the categories learned during training can result in severe misclassification outcomes, a significant issue commonly referred to as the *OOD problem* [17].

A prevalent yet somewhat limited strategy to tackle the OOD problem involves categorizing data not belonging to any known classes in the training set as the "others" class. The idea is that when faced with OOD data, the model would assign it to the "others" category. During training, samples from different distributions than the training data, such as random noise or data from alternate sources, are gathered and labeled as "others" to enable the model to grasp the features of this class. During inference, if an OOD sample shares characteristics with those in the "others" class, the model is likely to classify it accordingly. As depicted in Fig. 1, an experiment was conducted using CIFAR-10 as the ID dataset and SVHN as the "others" class for OOD data. However, while an "others" class may be defined, the real-world environment distribution of OOD data is virtually infinite, rendering it unfeasible to encompass all scenarios with limited training data. This limitation implies

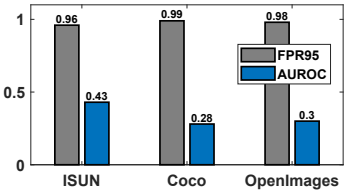


Fig. 1. Performance on different OOD datasets after training with CIFAR-10 as the ID data and SVHN as the "*others*" class. Lower FPR95 and higher AUROC are better.

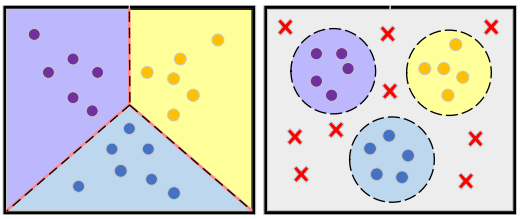


Fig. 2. Assuming a three-class scenario, the left image illustrates a standard classification model, while the right image depicts the feature representations after fine-tuning with OOD samples.

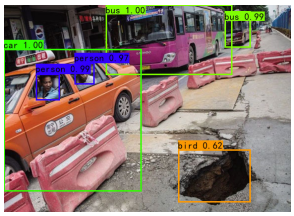


Fig. 3. Classification models may sometimes make erroneous predictions in typical road scenarios, such as misclassifying a pothole as a bird.

that in practical applications, the model may encounter unseen OOD samples that significantly differ from the "others" class, potentially leading to misclassification.

2.2 Motivation: OOD Detection and Beyond

To effectively address the OOD problem, the identification of OOD data is crucial to prevent the application of classification models to such data. Generally, OOD detection can be framed as a binary classification task, determining whether the input object belongs to the ID or OOD category. Given a test input $x^* \sim P_{\mathcal{X}}$ and the predicted bounding box b^* by the object detector, the objective is to predict $p_\theta(g|x^*,b^*)$, where $g = 1$ denotes that the detected object is ID and $g = 0$ indicates that it is OOD.

Numerous research efforts have focused on the problem of out-of-distribution (OOD) detection. For example, Softmax-based methods [7], [8] are simple and efficient but often suffer from overly confident decision boundaries formed during neural network training. As shown in Fig. 2 (left), OOD samples far from the decision boundary can still be misclassified into known categories due to high-confidence Softmax outputs, leading to detection failures. Generative model-based methods [9], [10], [18] aim to detect OOD samples by modeling the true data distribution. However, training generative models on high-dimensional data is often unstable, and the quality of generated samples is limited by the model's capacity. Additionally, generative models may assign high likelihood scores to OOD data (e.g., likelihood estimation anomalies in flow-based models), resulting in misclassifications. In addition, Distance-based methods [11], [12] utilize intermediate neural network features for OOD detection by calculating distances (e.g., Mahalanobis distance) between samples and the known category distribution. These methods rely heavily on the network's feature representation. If the extracted features do not accurately capture class structures or are influenced by noise, detection performance can significantly decline. Furthermore, covariance estimation in feature space can be unstable, especially in small-sample or high-dimensional settings, leading to distorted distance metrics.

Decision boundary-based methods [13], [14], [19] improve OOD detection by generating virtual OOD samples near the classifier's decision boundary to optimize performance. As depicted in Fig. 2 (right), training with OOD samples close to the decision boundary results in a more compact and precise boundary. These methods effectively address issues of overconfidence in Softmax-based approaches, likelihood confusion in generative models, and feature quality dependence in distance-based methods. However, they face challenges in obtaining OOD data. Although synthetic OOD samples can augment the dataset, they may not fully represent the diversity of real-world OOD data, which can include features or variations not present in the training set. Consequently, even with virtual OOD samples, the decision boundary may have limitations in generalizing to novel OOD data. Additionally, as time progresses and environments change, new OOD samples may continuously emerge, complicating the adaptability of methods reliant solely on pre-trained decision boundaries.

In addition to algorithmic challenges, Many existing OOD detection methods face substantial limitations in practical deployment, especially on resource-constrained mobile edge devices. These methods are usually developed and evaluated under resource-rich, server-grade conditions [14], with limited consideration of the resource constraints such as memory availability and energy efficiency [20]. In particular, state-of-the-art methods often depend on large backbone networks and complex multi-stage post-processing modules [21], leading to significant memory consumption and high-power demands. For instance, softmax- and distance-based models typically consume more than 1GB of GPU memory at inference time and impose considerable energy overhead during inference [22]. Generative approaches (e.g., VAEs or normalizing flows) further exacerbate this issue due to their high-dimensional representations and intricate architectures [23]. Without overcoming these resource bottlenecks, these existing OOD methods are difficult to be applied in practice, especially in latency- and energy-sensitive scenarios, such as autonomous driving, mobile vision, and industrial IoT.

To address these challenges, our `POD` framework dynamically adjusts decision boundaries during inference based on newly observed OOD samples. This adaptive mechanism enhances detection accuracy and system robustness in open-world environments. To further ensure efficient deployment, instead of relying on deep and computationally expensive architectures, `POD` applies a targeted pruning strategy based on gradient similarity analysis to effectively eliminate redundant parameters and non-contributory layers. It preserves only the most important components required for reliable OOD detection, thereby enabling efficient inference without compromising detection accuracy, suitable for deployment on mobile edge devices.

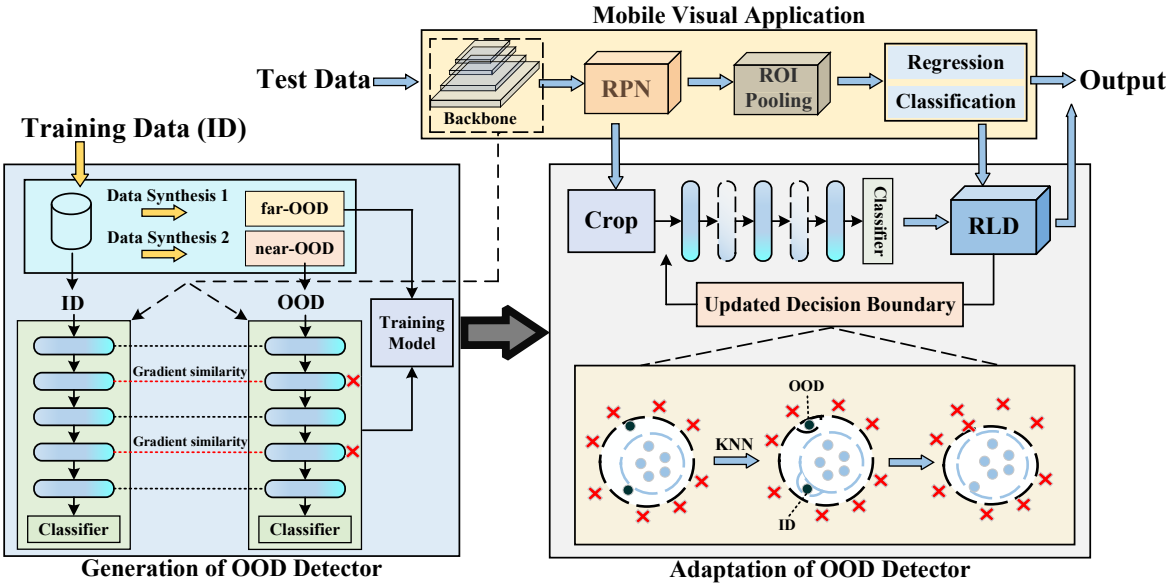


Fig. 4. Framework of POD.

2.3 POD in Real-World Applications

Due to its superior performance, POD can support a wide range of real-world applications as below.

- Autonomous driving systems operate in complex and dynamic environments, where onboard sensors may capture scenes unseen during training, such as extreme weather, rare traffic signs, or unexpected obstacles. Detecting OOD samples enables the system to adopt conservative control strategies or alert the driver, enhancing safety. For example, Fig. 3 shows a pothole misclassified as a bird, illustrating the potential risks of OOD-induced errors.
- In medical imaging, variations in equipment, patient demographics, or emerging pathologies can cause substantial distribution shifts. POD can flag OOD samples from new hospitals or devices, prompting manual review and improving diagnostic reliability and patient safety.
- In industrial manufacturing, vision-based inspection systems may encounter defects caused by environmental changes or material variability. POD can promptly detect novel anomalies, allowing early intervention to maintain production quality and prevent defective products from reaching the market.

3 Overview

In this paper, we present an lightweight OOD detection framework – POD that is tailored for mobile visual applications. Fig. 4 illustrates the framework of POD, offering a comprehensive view of the workflow. POD consists of three modules, which are briefly introduced as follows.

Mobile visual application: POD can work with any mobile visual application. For simplicity, we exemplify POD with a standard two-stage object detection model [24]. Specifically, the backbone network of the mobile visual application processes test data to extract feature maps from the input image. The *region proposal network* (RPN) proposes candidate regions by sliding a smaller window across the feature map generated by the backbone, and then *non-maximum suppression* (NMS) is applied to reduce overlapping boxes, retaining the regions with the highest scores. The *region of interest* (ROI) pooling converts these varying-sized regions into fixed-size outputs. *Regression* refines the bounding boxes for a better fit around the objects, and finally *classification* determines the object's category within each bounding box.

Generation of OOD Detector: POD constructs a lightweight OOD detector through a one-time offline training phase. During this phase, in-distribution (ID) data is first used to generate corresponding OOD samples via a data synthesis strategy (i.e., Data Synthesis 2). Each ID/OOD data pair is then fed into two baseline models with identical architectures, where the baseline model corresponds to the backbone network used by the mobile visual application. If the mobile system operates as a black-box, we provide a reference backbone for this process.

Since the backbone network of the mobile visual application has been trained on ID data, it exhibits pronounced uncertainty when processing OOD inputs, resulting in stronger gradient responses during backpropagation. In contrast, a model that has not been trained on ID data is not optimized for any specific distribution, and thus exhibits relatively small gradient differences between ID and OOD samples. By inheriting the weights from the ID-trained backbone, this design effectively amplifies the gradient differences, enabling us to identify network layers that contribute minimally to OOD detection.

During forward propagation, we analyze the gradient similarity across layers in both baseline models and prune the components that contribute least to OOD discrimination. The retained structure forms an initial lightweight detector, though it does not yet possess OOD detection capabilities. We then fine-tune this detector using OOD samples generated from both Data Synthesis 1 and Data Synthesis 2,

employing an energy-based training strategy to endow the network with OOD discrimination ability. The final detector enhances the distinction between ID and OOD samples by learning the differences in energy distribution, allowing it to effectively separate the two based on energy scores. Further technical details will be elaborated in Section 4.

Adaptation of OOD Detector: During the online inference phase, we first extract regions of interest (RoIs) from the region proposal network and crop them to serve as input for the OOD detector. As discrepancies may arise between the predictions of the OOD detector and the mobile visual application, we introduce a Real Label Discriminator (RLD) to reconcile the outputs of both components. Specifically, the predictions from the OOD detector and the mobile application are jointly fed into the RLD. If the predicted label is deemed correct, the system adopts it as the final output; otherwise, the incorrect label is used to dynamically update the OOD detector.

In real deployment, the OOD detector can dynamically adjust its decision boundaries based on the characteristics of newly encountered OOD samples. As the system is exposed to an increasing variety of OOD instances, the decision boundary is continuously optimized, gradually enhancing the detector's ability to handle diverse OOD data. Further technical details will be elaborated in Section 5.

4 Plug-and-Play OOD Detector

In Section 4.1, we firstly acquire the baseline model and utilize a portion of synthesized OOD data to identify the most suitable weight parameters for OOD detection. Based on these parameters, an OOD detector is generated from the baseline model. In Section 4.2, we use another portion of the synthesized OOD data to train the OOD detector, thereby endowing the detector with basic OOD recognition capability. Finally, Section 4.3 provides technique details about our OOD data synthesis used in Sections 4.1 and 4.2.

4.1 Generation of Initial OOD Detector

We begin by extracting the backbone network of the original mobile visual application as a baseline. If the original model is closed-source, we use the MobileNetV3 model as an alternative baseline. The weights and architecture of the baseline model are typically optimized solely for ID data to achieve the best performance. To effectively determine the most important weight parameters for OOD detection, it is essential to establish a relationship between ID and OOD data. To this end, we pair each ID sample with a corresponding OOD sample, referred to as *near-OOD data*(as described in Section 4.3), which is as close as possible to the model's decision boundary.

Theoretically, *near-OOD data* corresponding to ID data may exhibit some degree of similarity. If we input both ID data and *near-OOD data* into two identical baseline models, and observe that a particular layer responds similarly to both types of data, we can infer that this layer has a relatively weak ability to distinguish between ID and OOD data. In such cases, pruning this layer would not significantly affect the OOD detection performance of the baseline model. This insight provides guidance for building an OOD detector: by pruning network layers that respond similarly to both ID and OOD data, we can optimize the computational efficiency of the model while retaining key discriminative capabilities. Although pruning may lead to a slight decline in the model's accuracy on ID data, this impact is manageable because, in our framework, the primary task of the OOD detector is to identify OOD data, while the classification of ID data is handled by the mobile visual application itself.

4.1.1 Identify candidate layers for the detector

We use the backbone network of the original model or the MobileNetV3 model as the baseline. First, we select two baseline models, f_{θ_1} and f_{θ_2}, both having identical design and parameters, i.e., $\theta_1 = \theta_2$, ensuring that any differences observed in subsequent comparisons are due to the variations in input data. Let x_{ID} represent the ID data input, and x_{OOD} represent the corresponding virtual OOD data input.

For each layer l, during the forward propagation process, we compute the gradient values for the inputs x_{ID} and x_{OOD} through the model f_θ, denoted as $g_t(x_{\text{ID}}, \theta) = \frac{\partial \mathcal{L}(f_\theta(x_{\text{ID}}))}{\partial \theta_l}$ and $g_t(x_{\text{OOD}}, \theta) = \frac{\partial \mathcal{L}(f_\theta(x_{\text{OOD}}))}{\partial \theta_l}$. Here, $\mathcal{L}$ represents the loss function, and θ_l denotes the parameters of layer l. To identify the layers most suitable for OOD detection, we update a counter based on the cosine similarity between the gradients. Finally, we select the top 50% of layers with the highest counter values as the candidates. The detailed steps and formulae are as follows:

$$\cos(g_l(x_{\text{ID}}, \theta), g_l(x_{\text{OOD}}, \theta)) = \frac{g_l(x_{\text{ID}}, \theta) \cdot g_l(x_{\text{OOD}}, \theta)}{\|g_l(x_{\text{ID}}, \theta)\| \|g_l(x_{\text{OOD}}, \theta)\|} \tag{1}$$

Here, $\cdot$ denotes the dot product operation, and $\|\cdot\|$ represents the norm of a vector. The cosine similarity of the gradients reveals how the model responds to ID data compared to the corresponding OOD data.

$$\text{if } \cos(g_t(x_{\text{ID}}, \theta), g_t(x_{\text{OOD}}, \theta)) \geq \delta, \quad C_l = C_l + 1 \tag{2}$$

We define a counter C_l to record the similarity occurrences for each layer l. Initially, all counters C_l are set to 0, i.e., $C_l = 0 \quad \forall l \in \{1, 2, \ldots, L\}$, where L denote the total number of layers. When the cosine similarity exceeds a certain similarity threshold δ, the counter C_l for layer l is incremented by 1 (we discuss the setting of δ in the experimental section). We select the top 50% of layers l with the highest counter values C_l as the candidates. Let L' represent the set of pruning candidate layers: $L' = \{l \mid l \in \text{top 50\% of } \{C_1, C_2, \ldots, C_L\}\}$. The final set of candidate layers can be represented as:

$$L' = \left\{ l \mid l \in \arg\max_{l \in \{1,\ldots,L\}} \left(\sum_{i=1}^{k} C_i \right), k = \lceil 0.5L \rceil \right\} \tag{3}$$

where $\lceil 0.5L \rceil$ denotes the ceiling function, which rounds up to ensure that we select the top 50% of the layers. This method allows us to effectively identify the layers most suitable for OOD detection based on gradient similarity. Finally, based on the parameters and layers identified, we generate the initial OOD detector from the baseline model.

4.1.2 Model compression

During the process of generating OOD detector from the baseline model, we can also apply various levels of model compression depending on the storage capacity of mobile device hosting the baseline model. When deriving the detector from the baseline model, we employ L1 norm-based pruning, defined as: $\|W_l\|_1 = \sum_{i=1}^{n} \sum_{j=1}^{m} |W_{l,ij}|$, where W_l represents the weight matrix of the l-th layer, and $W_{l,ij}$ denotes the element in the i-th row and j-th column of the weight matrix.

For each layer l, we generate an $L1$ pruning mask M_l. This mask is created by setting a threshold τ_l, pruning the elements of the weight matrix whose absolute values are smaller than this threshold. This can be expressed as:

$$M_{l,ij} = \begin{cases} 1 & \text{if } |W_{l,ij}| \geq \tau_l \\ 0 & \text{if } |W_{l,ij}| < \tau_l \end{cases} \tag{4}$$

where $M_{l,ij}$ is the pruning mask for the element $W_{l,ij}$. Finally, we update the weights using the pruning mask M_l: $W_l' = W_l \circ M_l$, where $\circ$ denotes the element-wise multiplication operation. This process retains only the significant weights in the l-th layer by zeroing out those weights whose values are below the specified threshold τ_l, as determined by the $L1$ norm pruning. Elements in the weight matrix W_l with absolute values below the threshold τ_l are effectively pruned. For devices with varying storage capacities, we can select different values of τ_l.

4.2 Enabling OOD Detection

After generating the initial OOD detector from the baseline model, we then enable its OOD detection capabilities by using a combination of synthesized *near-OOD data* and *far-OOD data* as training input, as illustrated in Fig. 4. Typically, OOD detection relies on a scoring function to distinguish between ID and OOD data. To this end, we employ an energy-based scoring method for fine-tuning the model. Energy-based models are inherently connected to discriminative models. The softmax discriminant function derives the classification distribution as follows:

$$p(y|x) = \frac{p(x,y)}{p(x)} = \frac{e^{f_y(x;\theta)}}{\sum_{i=1}^{K} e^{f_i(x;\theta)}} \tag{5}$$

where $f_y(x;\theta)$ denotes the y-th element of logit output corresponding to the label y. Taking the negative logarithm of p(x) in Equation (5) yields: $-logp(x) = -\log \sum_{i=1}^{K} e^{f_i(x;\theta)}$. The resulting function is referred to as the energy function $E(x;\theta)$, and it has been proven to be an effective uncertainty measurement for OOD detection [25].

$$E(x;\theta) = -\log \sum_{i=1}^{K} e^{f_i(x;\theta)} \tag{6}$$

where x represents the input image. $f_i(x;\theta)$ is the logit score for the i_{th} category of the target, K is the total number of categories, and θ represents the model's parameters. The model trained with the energy function will reduce the energy of ID data points [26]. When OOD samples are fed into the model, since they differ from the ID training data, the model might struggle to assign them a clear category, leading to an increase in energy values. During OOD detection, it is expected that the model assigns a lower energy value for ID data and a higher energy value for OOD data. Therefore, the model distinguishes ID and OOD data by their energy values, forming a basic OOD detector.

4.3 Virtual OOD Data Synthesis

The decision boundary is the boundary in the feature space that separates different classes in a classification model. By incorporating OOD data during training, the model's decision boundary can be adjusted to effectively distinguish between ID data and OOD data. However, if OOD data are selected directly from a dataset that is different from the training distribution, it may be difficult to capture the diversity near the decision boundary, thereby affecting the model's generalization capability. Existing methods for generating OOD data typically rely on natural perturbations (e.g., blurring, noise, or geometric transformations) or adversarial perturbations to generate samples close to the decision boundary of the classification model. However, these approaches often perform poorly on real-world environment OOD datasets, as the synthetic images tend to visually resemble ID samples. When confronted with natural OOD images that are farther away from the ID samples, the behavior of the model remains uncertain.

Therefore, it is essential to train the model using OOD samples that cover as much of the feature space as possible. PixMix [27] takes advantage of the natural structural complexity of images, such as fractals, to design a data augmentation method that generates complex and diverse OOD samples. This approach improves the performance of OOD detection without compromising the accuracy of the classification, outperforming most baseline methods. Furthermore, research [28] suggests that in the absence of knowledge of the OOD distribution, the optimal strategy is to uniformly select OOD samples from the feature space. Inspired by this theory, we define two types of OOD data: *near-OOD data* and *far-OOD data*. *Near-OOD data* refer to samples close to the model's decision boundary, belonging to the OOD distribution but lying in regions of high uncertainty; *far-OOD data*, on the other hand, are samples that are significantly distant from the decision boundary and deviate substantially from the training distribution. By combining both *near-OOD data* and *far-OOD data* for training, we can enhance both decision boundary accuracy and generalization.

In our framework, these two types of data are seamlessly integrated: *near-OOD data* is used as input during the pruning stage. After pruning, it is combined with *far-OOD data* for further training, thereby constructing an efficient OOD detector. This combination helps the model learn a more optimal decision boundary. Some existing data augmentation techniques can adequately fulfill our requirements for generating *near-OOD data* and *far-OOD data*. To generate *near-OOD data*, we apply various image corruption techniques [29], including Gaussian noise, elastic transformations, Gaussian blur, spatter, glass blur, and defocus blur. Since these modifications maintain a distribution relatively close to the original ID data in feature space, they are well-suited for *near-OOD data* representation. On the other

hand, CutMix [30] is employed to generate *far-OOD data* by cutting and replacing key regions between two images while blending their labels proportionally. The samples generated by CutMix exhibit significant divergence from the original ID data in feature space, making them highly suitable for representing *far-OOD data* samples. These techniques enable the construction of a detector with basic OOD recognition capabilities, which can be further optimized during testing.

The near-far OOD categorization not only enriches the diversity of OOD data used during training but also plays a crucial role in the POD framework. In our lightweight optimization technique, each training step requires an ID-OOD data pair, where the OOD sample is chosen from *near-OOD data* (i.e., samples generated through image corruption). The key reason for this choice is that we analyze the gradient similarity between ID and *near-OOD data* across different layers of the model. Since *near-OOD data* are closer to ID data in feature space, the layers showing significant gradient differences between ID and *near-OOD data* samples are the ones most critical for distinguishing OOD data.

5 Adaptation of OOD Detector

Due to the limited diversity of the synthesized OOD data, the resulting OOD detector may not be fully optimized. To address this issue, POD dynamically adjusts the OOD detector's decision boundaries during online use. Given the potential conflicts between the outputs of the OOD detector and the mobile visual application, we introduce a Real Label Discriminator (RLD) component into POD. The RLD is designed to discriminate labels and guide the dynamic adjustment of decision boundaries. In Section 5.1, we provide a detailed explanation of the decision boundary. Section 5.2 elaborates the dynamic adjustment of the decision boundary. Finally, Section 5.3 details the RLD design.

5.1 Decision Boundaries of OOD Detector

We define two types of decision boundaries for an OOD detector: the *internal decision boundary* B_{in} and the *external decision boundary* B_{out}. The internal decision boundary is defined based on the entire ID training dataset, as explained in Section 5.2. All outputs within this boundary are considered as ID data. The external decision boundary is derived from training with synthesized OOD data, representing the decision boundary established by the OOD detector during training. Any output from the OOD detector that falls outside this boundary is classified as OOD data.

Because the fixed decision boundary obtained from training (i.e., the external decision boundary B_{out}) may not always be accurate, there can be classification uncertainty if a data point x lies between the internal and external decision boundaries ($B_{in} < x < B_{out}$). This intermediate range creates a "gray area" where the data might belong to either the ID or OOD category. To more accurately classify data within this gray area, we introduce the concept of an energy score $E(x) = -\log\left(\sum_i e^{f_i(x)}\right)$ for a data point.

We employ the KNN algorithm to classify data based on energy score. Specifically, we have two distinct sets of energy scores: one set derived from ID data, denoted as $E_{\text{ID}} = \{e_1, e_2, \ldots, e_n\}$; and the other set derived from OOD data, denoted as $E_{\text{OOD}} = \{e'_1, e'_2, \ldots, e'_n\}$. We define the distance metric function for energy scores as follows: $d(E(x), E(y)) = \sqrt{\sum_{i=1}^{m} (E_i(x) - E_i(y))^2}$. For a given test data point x, we calculate its distance from all training data points: $D(x) = \{d(E(x), E(x_i)) \mid x_i \in \text{Training Data}\}$. We select the k nearest neighbors with the smallest distances: $N_k(x) = \{x_{(1)}, x_{(2)}, \ldots, x_{(k)} \mid d(E(x), E(x_{(1)})) \leq d(E(x), E(x_{(2)})) \leq \ldots \leq d(E(x), E(x_{(k)}))\}$, and then compute the weight for each neighbor as $w_i = \frac{1}{d(E(x), E(x_i)) + \epsilon}$. Finally, we predict the class label y for the new data point:

$$\hat{y} = \arg\max_{c \in \{0,1\}} \sum_{x_i \in N_k(x)} w_i \cdot \mathbb{1}(y_i = c) \tag{7}$$

where the indicator function $\mathbb{1}(y_i = c)$ takes the value of 1 if the label y_i of the neighbor x_i is equal to the class c, and 0 otherwise. We use label 0 and 1 to represent ID and OOD, respectively. This process ultimately determines the classification of samples that lie between the internal and external decision boundaries.

5.2 Adjustment of Decision Boundaries

In Section 5.1, we identify the class of samples that fall between the internal and external decision boundaries. This classification guides the adjustment of decision boundaries. During model testing, we only fine-tune for those data points that lie between the inner and outer decision boundaries, ensuring the model focuses its attention on those points where its recognition is in doubt. This approach helps the model more precisely handle edge cases without being influenced by its highly certain data points. At the same time, adjusting the outer decision boundary may have some impact on the inner decision boundary. However, in our framework, after each update of the outer decision boundary, the inner decision boundary is recalculated. This recalculation ensures that the inner decision boundary can adapt in real-time to changes in the outer decision boundary. The adjustment of the inner decision boundary only involves changes to the data points in the feature space and does not affect the outer decision boundary.

5.2.1 The internal decision boundary expands

When a new data point is determined to be ID data by the KNN algorithm but does not fall within the inner decision boundary, we label it as an extension point and moderately expand the inner decision boundary. This strategy aims to ensure that newly emerging, valid ID data can be correctly recognized by the model, thereby enhancing its generalization performance.

To expand the inner decision boundary, we employ a data energy score-based approach to determine and adjust the threshold. Initially, we establish an initial threshold based on the energy scores from the initial dataset. This threshold is subsequently used to determine whether data belongs to the ID category. The threshold setting method involves selecting a percentile of the energy scores:

$$\text{threshold} = \text{Percentile}(\{E(x_1), E(x_2), \ldots, E(x_N)\}, 95) \tag{8}$$

where $E(x_1)$, $E(x_2)$, ..., $E(x_N)$ represent the energy fractions of the initial ID data. The Percentile function is employed to derive the 95th percentile from these fractions. This computed value is subsequently designated as the internal decision boundary threshold. If the energy fraction of such data exceeds the existing threshold, a recalculation of the threshold is necessitated.

$$\begin{aligned}\text{if } \max(\{E(x_{\text{new}_1}), E(x_{\text{new}_2}), \ldots\}) > \text{threshold then}\\ \text{threshold} = \text{Percentile}(\{E(x_1), E(x_2), \ldots, E(x_N),\\ E(x_{\text{new}_1}), E(x_{\text{new}_2}), \ldots\}, 95)\end{aligned} \quad (9)$$

Here, the recalculation involves computing the 95th percentile of the energy fraction set, which includes both new and old data, in order to update the threshold.

5.2.2 The external decision boundary contracts

When a data point is identified as OOD data but is within the outer decision boundary, the outer boundary should be reduced to ensure that the OOD detector remains highly sensitive to genuine OOD data and reduces over-reliance on such data. This targeted strategy, which adjusts only for data lying between the two decision boundaries, effectively prevents frequent and unnecessary adjustments to the decision boundary, ensuring the OOD detector's robust operation.

To more effectively narrow down the outer decision boundary, we fine-tune the model in each test batch using energy scores. This is done to enhance the OOD detector's ability to handle OOD samples. Specifically, we calculate an energy score $E(x)$ for each data point x. At the testing phase, we utilize the energy score loss to optimize the OOD detector with the aim of generating higher energy scores for OOD samples. In each testing batch, we randomly select a small portion of ID data, denoted as $X_{\text{ID}}^{\text{batch}}$, and the current OOD testing data, denoted as $X_{\text{OOD}}^{\text{batch}}$, for OOD detector updating. We define a batch of ID and OOD data as follows: $X_{\text{ID}}^{\text{batch}} = \{x_{i_1}^{\text{ID}}, x_{i_2}^{\text{ID}}, \ldots, x_{i_b}^{\text{ID}}\}$, $X_{\text{OOD}}^{\text{batch}} = \{x_{j_1}^{\text{OOD}}, x_{j_2}^{\text{OOD}}, \ldots, x_{j_b}^{\text{OOD}}\}$. Where b is the batch size, $i_1, i_2, \ldots, i_b$ are random indices from the ID data, and $j_1, j_2, \ldots, j_b$ are random indices from the OOD data. Considering the randomness in batch selection and the optimization of the loss function, our objective is to minimize the following loss function:

$$\begin{aligned}L_{\text{batch}} = &\frac{1}{b}\sum_{i\in X_{\text{ID}}^{\text{batch}}}\left(\max(0, m_{\text{ID}} - E(x_i))\right)^2\\ &+\frac{1}{b}\sum_{j\in X_{\text{OOD}}^{\text{batch}}}\left(\max(0, E(x_j) - m_{\text{OOD}})\right)^2\end{aligned} \quad (10)$$

where the batch size b serves as a parameter used to control the amount of data utilized in each update. The total loss function for the current batch is represented as L_{batch}. The threshold m_{ID} represents the typical upper limit of the energy score for ID data, as described in Section 5.2.1, and is derived through calculation. Conversely, m_{OOD} denotes the typical lower limit of the energy score for OOD data. We have determined that setting the threshold at -3 is optimal for m_{OOD}. By optimizing the loss function L_{batch}, the OOD detector can fine-tune itself in each batch based on the energy scores of ID and OOD data. This approach effectively narrows the external decision boundary, enhancing the model's accuracy in recognizing OOD data.

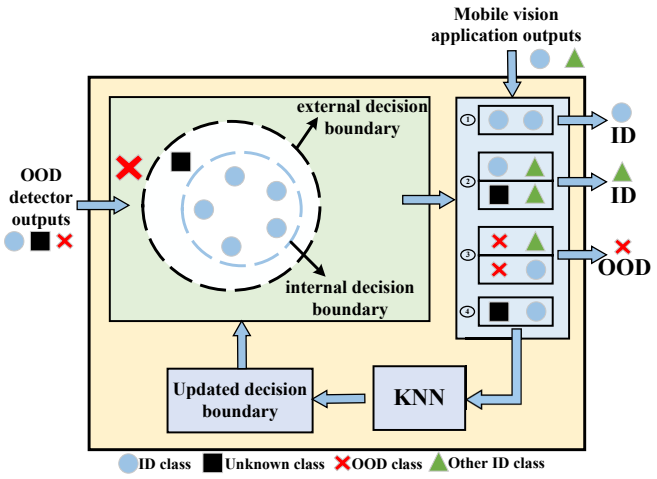


Fig. 5. RLD operates based on the decision rules defined in Section 5.3. For items 1, 2, and 3, the output can be directly determined. For item 4, KNN is used to determine the true class of the OOD detection output, followed by a dynamic update of the decision boundary.

During the training process, updates to the outer decision boundary can influence the energy distribution of the ID data. This is because the optimization of the outer decision boundary changes the energy distribution of the OOD data (e.g., ensuring $E(x_j) > m_{OOD}$), which in turn alters the parameters of the feature extraction layer. Since the feature extraction layer is shared, these adjustments inevitably affect the feature distribution of the ID data, thereby influencing the inner decision boundary. To ensure the correctness of the inner decision boundary, we recalculate it after each update to the outer decision boundary. On the other hand, updates to the inner decision boundary do not alter the parameters of the feature extraction layer. As a result, changes to the inner decision boundary do not have a reverse impact on the outer decision boundary.

5.3 Real-Label Discriminator

By analyzing and contrasting the outputs from both the OOD detector and mobile visual application, the RLD component aims to accurately determine the true class of the test data. As shown in Fig. 5, the detector is capable of categorizing data as either ID or OOD, whereas the mobile visual application is limited to producing ID classifications. Given these differences, the RLD must reconcile the outputs from both models to determine the correct classification. To achieve this, we consider all possible output combinations from the OOD detector and mobile visual application, allowing to generate reliable and precise classification results.

5.3.1 When OOD detector outputs ID and mobile visual application outputs ID

Both output the same: Since mobile visual applications are unable to discriminate OOD data, and the decision boundary of the OOD detector may be flawed due to the lack of diversity in artificially synthesized OOD data, there

TABLE 1
Comparison of POD's OOD detection with baselines. The specialized baseline models are ResNet50 and WRN, while MobileNetV3 serves as the general baseline model. BDD and VOC are used as ID datasets, and OpenImages and COCO as OOD datasets. Note: ↑ indicates higher values are better, and ↓ indicates lower values are preferable. All values are presented as percentages, with bold numbers representing the best results.

ID dataset	Model	Method	OpenImages		COCO	
			FPR95↓	AUROC↑	FPR95↓	AUROC↑
VOC	ResNet50	MSP [8]	92.03	64.21	88.44	64.99
		Energy score [25]	67.63	79.47	65.13	79.03
		VIM [21]	58.17	83.75	56.43	82.52
		VOS [14]	46.37	86.78	44.48	86.97
		POD	**33.75**	**91.88**	**36.31**	**91.45**
	MobileNetV3	POD **(common)**	41.72	85.57	43.75	82.25
BDD	WRN	MSP [8]	80.63	75.84	79.78	76.09
		Energy score [25]	49.69	84.63	50.00	84.79
		VIM [21]	42.70	90.16	45.15	88.95
		VOS [14]	22.21	93.36	29.83	91.53
		POD	**15.31**	**95.80**	**17.50**	**95.91**
	MobileNetV3	POD **(common)**	26.88	91.23	31.56	90.61

is a possibility of erroneous judgments in their ID outputs. Hence, there are two possible output regions. The first is when the output is within the inner decision boundary of OOD detector; in this case, we consider the data as ID without further judgment. The second case is when the output is between the inner and outer decision boundaries of OOD Detector. For this case, we employ the KNN algorithm in Section 5.1 to further discern if the data is ID or OOD, and update the model's decision boundary accordingly.

Both output differently: As the mobile visual application has superior ability in processing ID data compared to the OOD detector, we thus consider the mobile visual application's output as more reliable. Under this circumstance, the classification error appearing in the OOD detector may be due to performance degradation from pruning. As illustrated in Fig. 5, when the OOD detector exhibits an ID classification error, it typically implies that the output falls on the boundary between two ID categories. In this case, we believe that the output is not OOD data, because if it was OOD data, its output would be distant from the centers of these two categories, leaning more towards the outer decision boundary. Therefore, we directly adopt the output from the mobile visual application as the final result.

5.3.2 When OOD detector outputs OOD and mobile visual application outputs ID

The mobile visual application inherently lacks the capability to distinguish OOD, whereas the OOD detector is more accurate in its judgment of OOD. Therefore, when the OOD detector perceives the output as OOD, and this output is outside the outer decision boundary, we will directly categorize it as OOD, without taking into consideration the output from the mobile visual application.

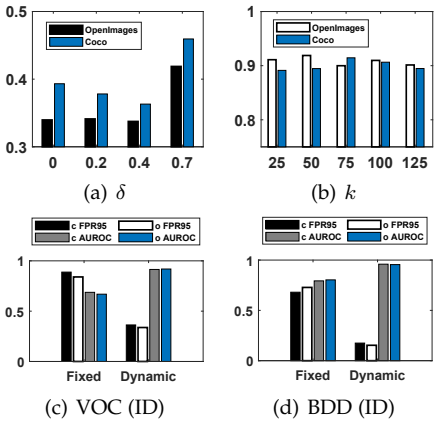
6 IMPLEMENTATION

We implement our approach using NVIDIA CUDA 11.8 and Python 3.9. The experiments are conducted on an NVIDIA GeForce RTX 4070, where we perform a range of evaluations. Additionally, we test OOD detectors generated with specialized models on the XiaoMi 14 and those generated with general models on the Raspberry Pi 4B.

Models and datasets: We evaluate POD using the widely adopted Faster R-CNN model [24], which is commonly used in visual applications. The backbone networks of Faster R-CNN model can serve as baseline models for generating OOD detectors, including ResNet50 [31], WRN [32], and MobileNetV3 Small [33]. MobileNetV3 is utilized as a general model for various applications, while ResNet50 and WRN are employed as application-specific models for scenarios such as road monitoring and autonomous driving. We train and test these models on the following datasets:

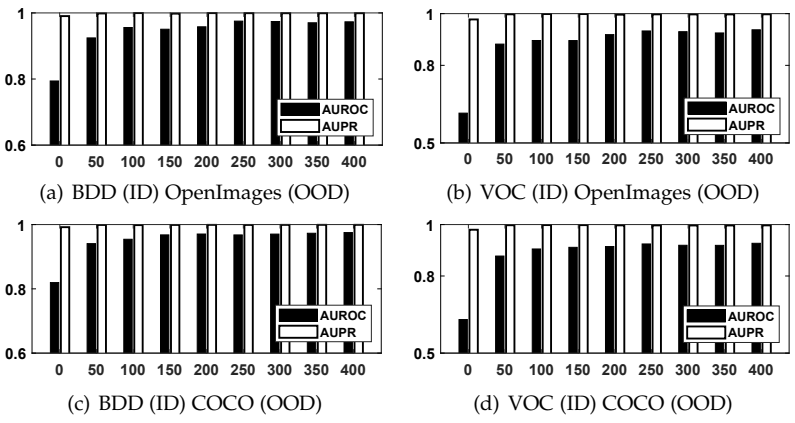
1) **PASCAL VOC [34]** comprises a series of images and the corresponding annotations, covering 20 different object categories, such as humans, animals, vehicles, and furniture. It will be used for ID data training.
2) **Berkeley DeepDrive [35]** is a large-scale, diverse, and rich autonomous driving dataset containing 100,000 driving video clips. This dataset contains 12 object categories, such as vehicles, pedestrians, and traffic lights. It will be used for ID data training.
3) **MS COCO [36]** includes 80 object categories. It features a wide range of everyday objects such as utensils, electronics, furniture, and animals. The substantial distributional difference between COCO and VOC/BDD makes it an ideal OOD dataset.
4) **OpenImages [37]** contains hundreds of object categories, including household items, animals, scene elements, and abstract objects. This broad coverage makes it a highly challenging OOD dataset.

System training: During the training phase of POD, we first extract the backbone network from the mobile visual application as the base model for the initial OOD detector. This model is then lightweighted and trained using synthetic OOD data to enable OOD recognition capabilities. We implemented the full training pipeline in PyTorch and conducted all model training on a single NVIDIA GPU. The ID datasets are sourced from VOC and BDD, from which we crop 224×224 images for training. The training batch size was set to 32. To evaluate the generalizability of our method, we use WideResNet (WRN) and ResNet-50 as backbone networks. WRN is configured as WRN-40-2 (40 layers, widening factor of 2) with a dropout rate of 0.3. Training



(a) δ (b) k (c) VOC (ID) (d) BDD (ID)

Fig. 6. Impact of (a) variance δ, (b) parameter k, and (c-d) dynamic decision boundary adjustment.



(a) BDD (ID) OpenImages (OOD) (b) VOC (ID) OpenImages (OOD) (c) BDD (ID) COCO (OOD) (d) VOC (ID) COCO (OOD)

Fig. 7. Performance (AUROC and AUPR) and latency (unit: ms) variation with increasing OOD samples. The baseline models for VOC and BDD are ResNet50 and WRN, respectively.

is performed using stochastic gradient descent (SGD) for 50 epochs, with a momentum of 0.9, weight decay of 0.0005, an initial learning rate of 0.001, and cosine annealing learning rate scheduling.

Metrics: To evaluate OOD detection performance, we report: (1) the false positive rate (FPR95) of OOD samples when the true positive rate of ID samples is 95%; (2) the area under the receiver operating characteristic curve (AUROC); and (3) the area under the precision-recall curve (AUPR).

7 Experiments

7.1 Overall Performance

We firstly investigate the overall OOD detection performance of four different methods. The compared baselines include Maximum Softmax Probability, Energy score, and VIM. The MSP method classifies samples by calculating the maximum value of the softmax probabilities. The Energy Score approach, on the other hand, takes logits as input and maps them using an energy function. VIM, built on a Transformer architecture, integrates amplitude and distributional information of features, effectively capturing internal model variations when processing diverse samples. For the same dataset (application), we utilize identical baseline models. Additionally, we provide a general version for each dataset (application). For the experimental results, a lower FPR95 and a higher AUROC are preferred. We employ BDD and VOC as ID datasets and train the models with our synthesized virtual data as OOD datasets. The models are then tested on OpenImages and COCO as OOD datasets.

The experimental results shown in Table 1 indicate that the specialized `POD` achieves the best performance across all scenarios. Moreover, `POD` enhances performance through an adaptive mechanism that dynamically adjusts the decision boundary of OOD detector. We compare `POD` with the VOS method, which employs synthetic virtual outliers for regularization to adjust decision boundaries, outperforming most baseline methods. Compared to VOS, our data synthesis method is more lightweight, albeit with fewer types of synthesized OOD data. By integrating dynamic decision boundaries, we further improved performance. `POD` demonstrated an 12.33% and 12.62% improvement in OOD detection performance (FPR95) on BDD (with COCO as OOD) and VOC (with OpenImages as OOD), respectively.

TABLE 2
The impact of OOD pruning rate on performance.

ID	Pruning rate	OpenImages		COCO	
		FPR95↓	AUROC↑	FPR95↓	AUROC↑
VOC	0%	41.22	89.74	45.69	88.83
	25%	39.34	89.51	42.19	90.63
	50%	**33.75**	**91.88**	**36.31**	**91.45**
	75%	35.28	90.08	38.75	90.81
BDD	0%	20.00	95.15	24.69	94.70
	25%	**12.81**	**96.59**	19.06	95.23
	50%	15.31	95.80	**17.50**	**95.91**
	75%	19.69	95.03	21.88	94.38

7.2 Micro-benchmarks

Impact of pruning rates. We evaluate the performance of the generated OOD detector across four pruning rates: 0%, 25%, 50%, and 75%. As shown in Table 2, the results indicate that pruning not only maintains but even enhances the model's OOD performance. This improvement can be attributed to our strategy for generating the OOD detector, which preserves the most critical parameters for OOD detection while eliminating less relevant ones. Notably, at a pruning rate of 50%, the model achieves a lightweight architecture without compromising OOD accuracy. Therefore, we adopt a 50% pruning rate in the final implementation.

Impact of variance δ. The appropriate setting of δ is crucial in determining the candidate layers for pruning. We use ResNet50 as the baseline model and VOC as the ID dataset, and report the changes in FPR95 across two OOD datasets: COCO and OpenImages. Equation (1) indicates that candidate layers are marked when the cosine similarity exceeds δ. In Fig. 6 (a), we analyze the impact of δ on OOD detection performance. Specifically, $\delta = \{0, 0.20, 0.40, 0.75\}$. We observe that when δ is below 0.50, the model's performance is generally higher. However, in extreme cases where δ is 0.75—meaning that only layers with an ID to OOD similarity greater than 0.75 are marked—some layers might not be marked at all. This results in subsequent

TABLE 3
The impact of different architectures on POD.

Architectures	ID Dataset	OpenImages		COCO	
		FPR95	AUROC	FPR95	AUROC
FasterRCNN	VOC	33.75	91.88	36.31	91.45
DERT		38.43	90.17	40.68	89.91

TABLE 4
The impact of different data augmentation methods on the OOD detection performance.

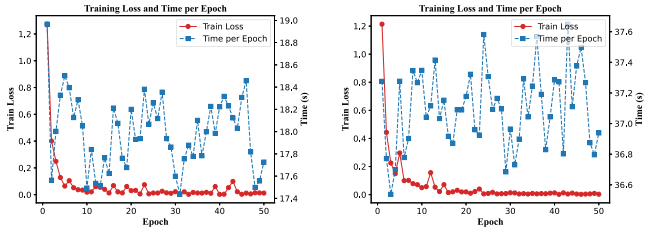
ID	Method	OpenImages		COCO	
		FPR95↓	AUROC↑	FPR95↓	AUROC↑
VOC	far-OOD [30]	**39.78**	84.77	51.56	77.34
	near-OOD [29]	48.47	79.59	47.97	82.19
	Pixmix [27]	44.18	83.64	44.75	80.39
	Our method	41.72	**84.79**	**43.75**	**82.25**
BDD	far-OOD [30]	30.63	85.84	32.50	87.35
	near-OOD [29]	30.61	89.97	34.81	85.54
	Pixmix [27]	28.69	86.83	**29.06**	86.15
	Our method	**26.88**	**91.23**	31.56	**90.61**

random pruning, which can lead to a decline in the detection performance.

Impact of k in KNN algorithm. We analyze the impact of k, the number of nearest neighbors, on determining the class of test data. We use ResNet50 as the baseline model and VOC as the ID dataset, and report the changes in FPR95 across two OOD datasets: COCO and OpenImages. Specifically, we vary k as 25, 50, 75, 100 and 125. Results in Fig. 6 (b) indicate that our method is not sensitive to this hyperparameter. The model's performance fluctuates without a discernible pattern across the range of 25 to 125. Therefore, we select $k = 75$ as it yielded the best result.

The impact of different architectures on POD. In our experiments, we compared the Transformer-based DETR [38] architecture with the conventional Faster R-CNN object detection framework. Since the inputs to POD are cropped image regions derived from the anchor boxes generated and selected by these object detectors after feature extraction, the differences in localization accuracy and quality of the proposed regions across detection architectures can influence the performance of the POD detector.

As shown in Table 3, we observed that DETR yields slightly lower precision on the OOD datasets compared to Faster R-CNN. This is primarily due to the large number of small- to medium-sized objects (e.g., birds, dogs) in VOC dataset, for which DETR's bounding boxes tend to be less precise. Loose or misaligned boxes may introduce excessive background or truncate parts of the object in the cropped region, which can confuse the OOD classifier and result in more entangled feature distributions between ID and OOD samples. In contrast, Faster R-CNN's region proposal mechanism (RoI) is specifically designed to ensure high localization precision, making it more suitable for the cropping operation required by our method. Overall, the DETR + POD combination yields slightly lower detection accuracy compared to the Faster R-CNN + POD setup.



(a) BDD (b) VOC

Fig. 8. The time overhead during the OOD training phase based on the baseline model pre-trained on the BDD and VOC ID datasets.

7.3 Ablation study

Data augmentation for OOD. The quality of the synthesized OOD data is a critical factor influencing the performance of the generated OOD detector. First, it affects the process of generating the OOD detector from the baseline model; second, it impacts the detection performance during fine-tuning. In Table 4, We report experimental results based on our proposed concepts of "near-OOD" and "far-OOD" data, as well as OOD data generated using traditional methods. The results demonstrate that our data augmentation approach significantly reduces FPR95 in the POD. Unlike conventional methods, our approach addresses not only the model's adaptability to *near-OOD data* but also its performance on *far-OOD data*, thereby offering a more comprehensive solution.

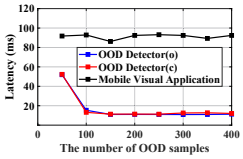
Impact of dynamic decision boundary. We train our model using BDD and VOC as ID datasets and evaluate its performance on OpenImages (o) and COCO (c) as OOD datasets. As shown in Fig. 6 (c-d), We conduct tests both with and without the dynamic decision boundary adjustment. Across different datasets, FPR95 significantly decreases, while AUROC improves. We find that, compared to fixed decision boundary methods, our dynamic decision boundary module substantially enhances OOD detection accuracy, with an average reduction in FPR95 by 52.70%.

POD's impact on memory usage and power consumption. As shown in Table 5, we further analyzed the impact of POD on memory usage and power consumption. We compared the memory and power consumption of the mobile visual application (MVA) when integrated with POD versus when operating without it. We selected MobileNetV3 as the baseline architecture.

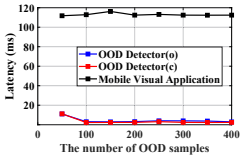
To better simulate inference scenarios on edge devices, we fixed the batch size to 1 while evaluating peak memory usage and power consumption, reflecting the inference load for a single image. In this setup, integrating POD into the mobile visual application resulted in a 17% increase in peak GPU memory usage and a 48% increase in average GPU power consumption. The experiment demonstrates that the

TABLE 5
Peak memory usage and average GPU power of mobile visual applications with and without POD integration.

POD model	Deployment	Peak memory (MB)	Avg GPU (W)
Mobilenetv3	MVA	854.96	31.02
	MVA + POD	998.66	45.88

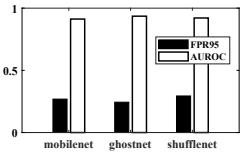


(a) BDD

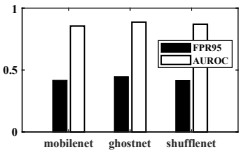


(b) VOC

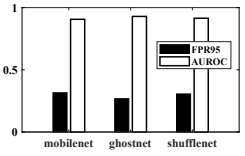
Fig. 9. The latency variations of POD with the increase in OOD samples.



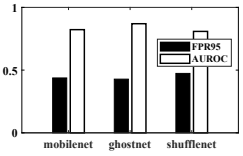
(a) BDD OpenImages



(b) VOC OpenImages

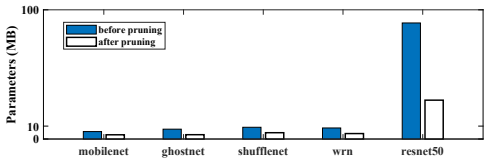


(c) BDD COCO

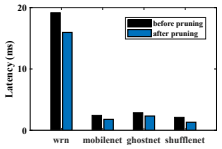


(d) VOC COCO

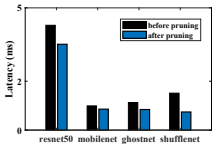
Fig. 10. The performance (FPR95 and AUROC) of MobileNet, GhostNet, and ShuffleNet as baseline models for OOD detection was evaluated.



(a) Model parameter memory usage



(b) BDD



(c) VOC

Fig. 11. (a) compares the memory usage of model parameters before and after pruning, while (b) and (c) illustrate the latency performance.

POD framework maintains high efficiency, consuming only limited system resources.

7.4 System Overhead

Training overhead and loss of the OOD detector. As shown in Fig. 8, we report the loss and time overhead of the OOD detector (MobileNetV3) during the training phase. We observe that the red curve (training loss) drops rapidly at the beginning and stabilizes within approximately 10 epochs. This indicates that the model successfully learns effective information for distinguishing between ID and OOD data, demonstrating good convergence and training efficiency. Although the model quickly adapts to the synthetic OOD data and acquires a certain level of OOD recognition capability, its detection accuracy may still degrade when faced with large amounts of emerging and diverse OOD categories in real-world environments. To address this, we introduce an adaptive decision boundary update mechanism during the testing phase to enhance the model's generalization ability in practical scenarios.

Meanwhile, the blue curve (training time per epoch) shows that the fluctuation in training time remains within 2 second per epoch. Although training times vary depending on the model size, the overall computational cost remains within an acceptable range.

Impact of test samples on performance. We use VOC and BDD as the ID training sets, and select portions of the COCO and OpenImages datasets as OOD test samples. As shown in Fig. 7, the performance of our model improves as the number of test samples increases. This improvement is primarily due to the dynamic adjustment and refinement of decision boundaries in response to OOD data that lies between the internal and external decision boundaries. Assuming accurate assessment of these test data, our decision boundaries will theoretically become more precise and robust as the number of test samples increases.

Impact of test samples on latency. As shown in Fig. 9, we report the average latency observed when processing 50 newly encountered OOD samples for the OOD detector and the mobile visual application across different datasets, OpenImages (o) and COCO (c). ResNet50 was used for training on the VOC dataset, while WRN was used for the BDD dataset. The black curve represents the average latency of the mobile visual application when processing every 50 samples. The red and blue curves represent the average latency of the OOD detector when processing every 50 OOD samples, including the additional delay caused by decision boundary updates.

In the early stages of testing, the OOD detector needs to adapt to the newly encountered OOD data, so the decision boundary undergoes multiple dynamic adjustments, leading to an increase in latency. However, as the model gradually adapts to this type of OOD data, the latency stabilizes, and the need for further boundary adjustments decreases. Experimental results show that in most cases, the average latency of the OOD detector remains below 40ms, significantly lower than the processing latency of the mobile visual application. Furthermore, decision boundary updates are only triggered when samples fall between the predefined internal and external decision boundaries.

Our experiment simulates an extreme scenario where all samples are OOD data. However, in real-world deployments, OOD samples are typically fewer. Hence, the decision boundary is usually updated before the next detection cycle, ensuring that real-time performance is not affected.

Model Selection in Black-Box Mobile visual Applications. We conducted experiments using VOC and BDD as ID datasets and COCO and OpenImages as OOD datasets. MobileNetV3-Small, ShuffleNet, and GhostNet were evaluated under these conditions. As shown in Fig. 10, the results demonstrated that MobileNet and GhostNet exhibited superior performance. In subsequent experiments, we further assessed memory usage and latency. Among the models, MobileNet demonstrated more comprehensive performance, broader applicability, and greater generalization capabilities. Therefore, in scenarios where the mobile visual application is treated as a black box, MobileNet is selected as the benchmark model for the OOD detector.

The impact of pruning on memory usage and latency. We compared the parameter memory usage of models before and after pruning, as shown in Fig. 11 (a). Post-pruning, the models can be efficiently deployed on edge devices. Fig. 11 (b) and (c) illustrate the average latency performance of different models on OOD detection samples

(OpenImages) before and after pruning, using VOC and BDD as ID training datasets. The results demonstrate that POD ensures millisecond-level processing.

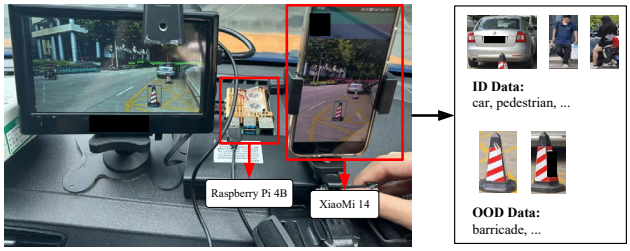
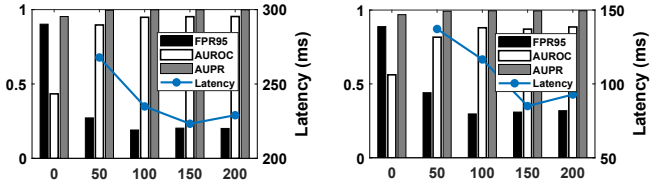


Fig. 12. Illustration of our real-world data collection using mobile devices.

Real-world environment testing. As shown in Fig. 12, we collect a real-world dataset in a road environment, capturing one frame per second, resulting in a total of 1,000 image frames as test samples. We use X-AnyLabeling [39] for image annotation and convert the annotations into the VOC dataset format for validation. Due to the limited amount of OOD data on the road, we select a portion of the LostAndFoundDataset [40] to serve as OOD data, so as to better validate the effectiveness of our method. The LostAndFoundDataset is a dataset used for obstacle detection in autonomous vehicles and intelligent transportation systems. It includes a variety of different types of obstacles, such as suitcases, trash bins, and rocks. These obstacles vary significantly in size, shape, and material to simulate the various situations that might be encountered in real driving environments.

We perform tests using both a Raspberry Pi 4B and a Xiaomi 14 smartphone. The Raspberry Pi 4B is equipped with a 1.5GHz quad-core ARM Cortex-A72 CPU, 4GB of LPDDR4-2400 SDRAM. The Xiaomi 14 is equipped with a Qualcomm Snapdragon 888 processor, 8GB of RAM, and 256GB of storage. On the Xiaomi 14, we employ Faster-RCNN for detection, utilizing its backbone network WRN as a specialized baseline model. On the Raspberry Pi 4B, assuming the detection model is unknown, we adopt MobileNetV3 as the general baseline model. As shown in Fig. 13, we conduct performance and latency tests in a real road environment. We observe that the model's performance improves progressively with an increasing number of test samples. However, due to the limited number of OOD samples, the improvement in OOD recognition performance is relatively modest. Additionally, the adjustment of decision boundaries allows the model to better adapt to OOD data, becoming increasingly refined. As a result, the frequency of decision boundary adjustments decreases, thereby reducing the processing latency.

We focus solely on evaluating the OOD detector's performance metrics, as the OOD detector is representative of the OOD detection capabilities within mobile visual applications. Moreover, when the mobile visual application and the OOD detector operate in parallel, the longer latency of the mobile visual application can effectively mask most of the latency introduced by the OOD detector. We find that both the smartphone and the Raspberry Pi maintain millisecond-level processing speed during the dynamic updates of decision boundaries, and their performance in real-world environments is also excellent.



(a) Xiaomi 14 (b) Raspberry Pi 4B

Fig. 13. Performance and latency variation with increasing test samples on different mobile devices.

8 Related Work

Mobile visual applications. Mobile visual applications have been widely deployed across edge platforms such as smartphones, in-vehicle systems, and wearable devices, supporting a broad range of tasks including image classification, object detection, scene understanding, and augmented reality [41]–[45]. The primary objective of these applications is to achieve efficient and accurate visual inference under resource-constrained environments. For instance, the MobileNet family and EfficientNet-Lite models [46]–[48] are commonly adopted for image classification due to their favorable trade-off between speed and accuracy, while lightweight variants of the YOLO series [49] enable millisecond-level object detection performance on mobile devices. In addition, numerous optimization techniques tailored for mobile platforms have been proposed, such as neural network pruning, quantization, and knowledge distillation [50]–[52], to further enhance runtime efficiency in embedded environments.

To improve the reliability of these systems in real-world scenarios, we propose POD, a lightweight OOD detector designed for automatic generation and dynamic adaptation. POD effectively strengthens the model's ability to recognize unknown inputs, thereby enhancing overall system robustness in practical open environments.

OOD detection. OOD detection has evolved significantly from early Softmax-based baselines, leading to a wide range of improved strategies. Specifically, Softmax-based methods [7], [8], [25], [53] aim to enhance OOD detection by post-processing the output probabilities of neural networks. For example, ODIN [7] applies temperature scaling and input perturbation to suppress the confidence scores of OOD samples, while GradNorm Score [53] further improves OOD discrimination by analyzing the gradient norm associated with the input. However, these methods are limited by the overconfidence of Softmax distributions and struggle to capture deep distributional characteristics of OOD samples at the feature level. In contrast, distance-based methods [11], [12], [54], [55] perform OOD detection directly in the intermediate feature space. Although these methods partially alleviate the overconfidence issue associated with Softmax, their performance heavily relies on the quality of feature representations, and distance metrics can become unstable in high-dimensional spaces.

Generative model-based approaches [9], [10], [18], [56] attempt to model the ID data distribution explicitly and

detect OOD samples by identifying anomalies in the generative space. Specifically, Generative Adversarial Networks (GANs) are used to assess OOD-ness based on discriminator confidence, while Variational Autoencoders (VAEs) are employed to detect anomalies using reconstruction errors or latent variable likelihoods. However, these approaches often suffer from training instability in high-dimensional data and may assign high likelihoods to certain OOD samples, as observed in flow-based models [57].

Recent efforts have focused on synthesizing virtual OOD samples near the decision boundary [13], [14], [19], [58] to enhance the classifier's discriminative capability. These methods bypass the complexities of modeling high-dimensional distributions found in generative models while improving the classifier's sensitivity to OOD data during training. However, the quality of synthetic OOD samples remains problematic, as they are often created through perturbations around decision boundaries and may not accurately reflect the complexity of real-world OOD data. To mitigate this issue, some approaches incorporate auxiliary OOD datasets, such as naturally occurring anomalous images [59] or GAN-generated samples [60], and use techniques like background-class regularization or the integration of unlabeled noisy data to improve detection performance. Additionally, various studies propose actively generating virtual OOD samples near the decision boundary. For example, VOS [14] maximizes sample entropy or classification uncertainty to guide the generator in producing challenging "fake" samples near the decision boundary. Similarly, BAL [61] employs adversarial learning to synthesize virtual OOD features, progressively generating harder samples to bolster the classifier's discriminative ability. In contrast, our POD framework adaptively adjusts the decision boundary during the testing phase, allowing it to effectively address the continuously emerging OOD data in real-world environments.

9 Conclusion

This paper presents POD, a framework designed to automatically generate corresponding OOD detectors for any classification model, enhancing the OOD detection capabilities of existing mobile visual applications without compromising their classification performance. POD operates across both training and testing phases: it generates OOD detectors during the training phase and continuously refines its performance in the testing phase by dynamically adjusting the decision boundaries. Extensive experiments demonstrate that POD significantly improves OOD detection performance with millisecond-level latency.

Acknowledgment

This work was supported in part by National Natural Science Foundations of China under Grant 62272388 and 62172284, and the Guangdong Provincial Key Lab of Integrated Communication, Sensing and Computation for Ubiquitous Internet of Things under Grant 2023B1212010007, and the GRF grants from Research Grants Council of Hong Kong (CityU 11205624 and 11202623).

References

[1] Y. M. et al., "Privacy-preserving face recognition using trainable feature subtraction," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 297–307.
[2] S. M. et al., "A contactless and non-intrusive system for driver's stress detection," in *ACM Int. Joint Conf. on Pervasive and Ubiquitous Computing and ACM Int. Symp. on Wearable Computing (UbiComp/ISWC '23 Adjunct)*, 2023, pp. 58–62.
[3] Y. K. et al., "Cityscouter: Exploring the atmosphere of urban landscapes and visitor demands with multimodal data," in *ACM Int. Joint Conf. on Pervasive and Ubiquitous Computing and ACM Int. Symp. on Wearable Computing (UbiComp/ISWC '23 Adjunct)*, 2023, pp. 157–161.
[4] J. N. et al., "Out-of-distribution detection for automotive perception," in *IEEE Int. Intell. Transp. Syst. Conf. (ITSC)*, 2021, pp. 2938–2943.
[5] X. Shen, Y. Wang, K. Zhou, S. Pan, and X. Wang, "Optimizing ood detection in molecular graphs: A novel approach with diffusion models," in *ACM SIGKDD Conf. on Knowledge Discovery and Data Mining (KDD)*, 2024, pp. 2640–2650.
[6] E. Nalisnick, A. Matsukawa, Y. Teh, D. Gorur, and B. Lakshminarayanan, "Do deep generative models know what they don't know?" in *Int. Conf. on Learning Representations (ICLR)*, 2019.
[7] Y.-C. Hsu, Y. Shen, H. Jin, and Z. Kira, "Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10 951–10 960.
[8] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," in *Int. Conf. on Learning Representations (ICLR)*, 2017.
[9] S. Mohseni, M. Pitale, J. B. S. Yadawa, and Z. Wang, "Self-supervised learning for generalizable out-of-distribution detection," in *AAAI Conf. on Artificial Intelligence*, vol. 34, no. 4, 2020, pp. 5216–5223.
[10] J. Serra, D. Álvarez, V. Gómez, O. Slizovskaia, J. F. Núñez, and J. Luque, "Input complexity and out-of-distribution detection with likelihood-based generative models," in *Int. Conf. on Learning Representations (ICLR)*, 2019.
[11] X. Du, G. Gozum, Y. Ming, and Y. Li, "Siren: Shaping representations for detecting out-of-distribution objects," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2024.
[12] Y. Li, X. Xu, Y. Su, and K. Jia, "On the robustness of open-world test-time training: Self-training with dynamic prototype expansion," in *IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2023, pp. 11 802–11 812.
[13] L. Tao, X. Du, J. Zhu, and Y. Li, "Non-parametric outlier synthesis," in *Int. Conf. on Learning Representations (ICLR)*, 2023.
[14] X. Du, Z. Wang, M. Cai, and Y. Li, "Vos: Learning what you don't know by virtual outlier synthesis," in *Int. Conf. on Learning Representations (ICLR)*, 2022.
[15] J. Li, Y. Wen, and L. He, "Scconv: Spatial and channel reconstruction convolution for feature redundancy," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 6153–6162.
[16] X. Chen, X. Wang, J. Zhou, Y. Qiao, and C. Dong, "Activating more pixels in image super-resolution transformer," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 22 367–22 377.
[17] J. Yang, K. Zhou, Y. Li, and Z. Liu, "Generalized out-of-distribution detection: A survey," *Int. J. Comput. Vis.*, vol. 132, no. 12, pp. 5635–5662, 2024.
[18] L. Neal, M. Olson, X. Fern, W.-K. Wong, and F. Li, "Open set learning with counterfactual images," in *Eur. Conf. on Computer Vision (ECCV)*, 2018, pp. 613–628.
[19] H. Zhang, H. Xu, and T.-E. Lin, "Deep open intent classification with adaptive decision boundary," in *AAAI Conf. on Artificial Intelligence*, vol. 35, no. 16, 2021, pp. 14 374–14 382.
[20] C. Lin, K. Wang, Z. Li, and Y. Pu, "A workload-aware dvfs robust to concurrent tasks for mobile devices," in *ACM Int. Conf. on Mobile Computing and Networking (MobiCom)*, 2023, pp. 1–16.
[21] H. Wang, Z. Li, L. Feng, and W. Zhang, "Vim: Out-of-distribution with virtual-logit matching," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 4921–4930.
[22] J. Liu, J. Liu, W. Du, and D. Li, "Performance analysis and characterization of training deep learning models on mobile device," in *IEEE Int. Conf. on Parallel and Distributed Systems (ICPADS)*, 2019, pp. 506–515.

[23] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks, "Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models," *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)*, vol. 44, no. 11, pp. 7327–7347, 2022.

[24] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)*, vol. 39, no. 6, pp. 1137–1149, 2017.

[25] W. Liu, X. Wang, J. Owens, and Y. Li, "Energy-based out-of-distribution detection," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 33, 2020, pp. 21 464–21 475.

[26] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. J. Huang, "A tutorial on energy-based learning," in *Predicting Structured Data*, 2006.

[27] D. H. et al., "Pixmix: Dreamlike pictures comprehensively improve safety measures," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 16 762–16 771.

[28] Z. Fang, J. Lu, A. Liu, F. Liu, and G. Zhang, "Learning bounds for open-set learning," in *Int. Conf. on Machine Learning (ICML)*, 2021, pp. 3122–3132.

[29] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," in *Int. Conf. on Learning Representations (ICLR)*, 2018.

[30] S. Yun, D. Han, S. Chun, S. J. Oh, Y. Yoo, and J. Choe, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2019, pp. 6022–6031.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[32] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *British Machine Vision Conf. (BMVC)*, 2016.

[33] A. H. et al., "Searching for mobilenetv3," in *IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2019, pp. 1314–1324.

[34] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *Int. J. Comput. Vis.*, vol. 88, pp. 303–338, 2010.

[35] F. Y. et al., "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2636–2645.

[36] T.-Y. L. et al., "Microsoft coco: Common objects in context," in *Eur. Conf. on Computer Vision (ECCV)*, 2014, pp. 740–755.

[37] A. K. et al., "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *Int. J. Comput. Vis.*, vol. 128, no. 7, pp. 1956–1981, 2020.

[38] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Eur. Conf. on Computer Vision (ECCV)*, 2020, pp. 213–229.

[39] W. Wang, "Advanced auto labeling solution with added features," https://github.com/CVHub520/X-AnyLabeling, 2023.

[40] P. Pinggera, S. Ramos, S. Gehrig, U. Franke, C. Rother, and R. Mester, "Lost and found: Detecting small road hazards for self-driving vehicles," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2016, pp. 1099–1106.

[41] M. Barz and D. Sonntag, "Gaze-guided object classification using deep neural networks for attention-based computing," in *ACM Int. Joint Conf. on Pervasive and Ubiquitous Computing: Adjunct*, 2016, pp. 253–256.

[42] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, and F. Kawsar, "An early resource characterization of deep learning on wearables, smartphones and internet-of-things devices," in *Proc. Int. Workshop on Internet of Things Towards Applications*, 2015, pp. 7–12.

[43] X. Zeng, K. Cao, and M. Zhang, "Mobiledeeppill: A small-footprint mobile deep learning system for recognizing unconstrained pill images," in *Int. Conf. on Mobile Systems, Applications, and Services (MobiSys)*, 2017, pp. 56–67.

[44] V. Radu, N. D. Lane, S. Bhattacharya, C. Mascolo, M. K. Marina, and F. Kawsar, "Towards multimodal deep learning for activity recognition on mobile devices," in *ACM Int. Joint Conf. on Pervasive and Ubiquitous Computing: Adjunct*, 2016, pp. 185–188.

[45] Z. Li, M. Li, J. Wang, and Z. Cao, "Ubiquitous data collection for mobile users in wireless sensor networks," in *IEEE INFOCOM*, 2011, pp. 2246–2254.

[46] A. G. H. et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[47] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520.

[48] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10 778–10 787.

[49] A. Nazir and M. A. Wani, "You only look once - object detection models: A review," in *Int. Conf. on Computing for Sustainable Global Development*, 2023, pp. 1088–1095.

[50] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," in *Int. Conf. on Learning Representations (ICLR)*, 2019.

[51] V. L. et al., "Manas: Multi-agent neural architecture search," *Mach. Learn.*, vol. 113, no. 1, pp. 73–96, 2024.

[52] B. J. et al., "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2704–2713.

[53] R. Huang, A. Geng, and Y. Li, "On the importance of gradients for detecting distributional shifts in the wild," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 34, 2021, pp. 677–689.

[54] V. Sehwag, M. Chiang, and P. Mittal, "Ssd: A unified framework for self-supervised outlier detection," in *Int. Conf. on Learning Representations (ICLR)*, 2021.

[55] J. Tack, S. Mo, J. Jeong, and J. Shin, "Csi: Novelty detection via contrastive learning on distributionally shifted instances," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020, pp. 11 839–11 852.

[56] C.-H. Lai, D. Zou, and G. Lerman, "Robust subspace recovery layer for unsupervised anomaly detection," in *Int. Conf. on Learning Representations (ICLR)*, 2020.

[57] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan, "Do deep generative models know what they don't know?" in *Int. Conf. on Learning Representations (ICLR)*, 2019.

[58] X. Du, X. Wang, G. Gozum, and Y. Li, "Unknown-aware object detection: Learning what you don't know from videos in the wild," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 13 678–13 688.

[59] D. Hendrycks, M. Mazeika, S. Kadavath, and D. Song, "Using self-supervised learning can improve model robustness and uncertainty," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.

[60] K. Lee, H. Lee, K. Lee, and J. Shin, "Training confidence-calibrated classifiers for detecting out-of-distribution samples," in *Int. Conf. on Learning Representations (ICLR)*, 2019.

[61] S. Pei, X. Zhang, B. Fan, and G. Meng, "Out-of-distribution detection with boundary aware learning," in *Eur. Conf. on Computer Vision (ECCV)*, 2022, pp. 235–251.

Zixiao Wang received the B.E. degree in Software Engineering from Lanzhou Jiaotong University in Gansu, China, in 2022. He is currently a Master's degree candidate in Software Engineering with the School of Information Science and Technology, Northwest University, Xi'an, China. His research interests include edge computing and mobile visual computing.

Qi Dong received the B.E. degree in Software Engineering from the School of Information Science and Technology, Northwest University, Xi'an, China, in 2024. His research interests include machine learning and large language models (LLMs).

Tianzhang Xing (Member, IEEE) received the B.E. degree in Telecommunications Engineering from Xidian University, Xi'an, Chian, in 2004, the M.Phil. degree and Ph.D. degree in computer science and technology from the Northwest University, Xi'an, China, in 2009 and 2014. He is currently an associate professor in the School of Information and Technology at Northwest University. His research interests include mobile computing, pervasive computing and wireless networks.

Zhidan Liu (Senior Member, IEEE) received the PhD degree in computer science and technology from Zhejiang University, Hangzhou, China, in 2014. After that, he worked as a research fellow with Nanyang Technological University, Singapore, and a faculty member with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. He is currently an assistant professor with Intelligent Transportation Thrust, System Hub, The Hong Kong University of Science and Technology (Guangzhou). His research interests include Artificial Internet of Things, mobile computing, urban computing, and Big Data analytic. He is a senior member of IEEE and CCF.

Zhenjiang Li (Member, IEEE) received the B.E. degree from Xi'an Jiaotong University, Xi'an, China, in 2007, and the M.Phil. and Ph.D. degrees from The Hong Kong University of Science and Technology, Hong Kong, in 2009 and 2012, respectively. He is currently an Associate Professor with the Department of Computer Science, City University of Hong Kong. His research interests include edge/embedded AI systems, Artificial Internet of Things (AIoT), and low-power systems.

Xiaojiang Chen (Member, IEEE) received the Ph.D. degree in computer software and theory from Northwest University, Xi'an, China, in 2010. He is currently a Professor with the School of Information Science and Technology, Northwest University. His current research interests include localization and performance issues in wireless ad hoc, mesh, and sensor networks and named data networks.