IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 6 January 2025, accepted 18 January 2025, date of publication 24 January 2025, date of current version 29 January 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3534020

RESEARCH ARTICLE

Knowledge Distillation in Object Detection for Resource-Constrained Edge Computing

ARIEF SETYANTO[1], (Member, IEEE), THEOPILUS BAYU SASONGKO[2], MUHAMMAD AINUL FIKRI[3], DHANI ARIATMANTO[1], I. MADE ARTHA AGASTYA[2], RAKANDHIYA DAANII RACHMANTO[4], AFFAN ARDANA[2], AND IN KEE KIM[4], (Member, IEEE)

[1]Magister of Informatics, Universitas Amikom Yogyakarta, Sleman 55283, Indonesia
[2]Department of Informatics, Universitas Amikom Yogyakarta, Sleman 55283, Indonesia
[3]Department of Informatics Engineering, Jember State Polytechnic, Jember 68101, Indonesia
[4]School of Computing, University of Georgia, Athens, GA 30602, USA

Corresponding author: Arief Setyanto (arief_s@amikom.ac.id)

This work was supported by the Ministry of Education, Culture, Research and Technology, Directorate General of Higher Education, under Grant No. 107/E5/PG.02.00.PL/2024.

ABSTRACT Edge computing, a distributed computing paradigm that places small yet capable computing devices near data sources and IoT sensors, is gaining widespread adoption in various real-world applications, such as real-time intelligent drones, autonomous vehicles, and robotics. Object detection (OD) is an essential task in computer vision. Although state-of-the-art deep learning-based OD methods achieve high detection rates, their large model size and high computational demands often hinder deployment on resource-constrained edge devices. Given their limited memory and computational power, edge devices like the Jetson Nano (J. Nano), Jetson Orin Nano (Orin Nano), and Raspberry Pi 4B (Raspi4B) require model optimization and compression techniques in order to deploy large OD models such as YOLO. YOLOv4 is a widely used OD model with a backbone for image feature extraction and a prediction layer. Originally, YOLOv4 was designed to use CSPDarkNet53 as its backbone, which requires significant computational power. In this paper, we propose replacing its backbone with a smaller model, such as MobileNetV2 and RepViT. In order to ensure the strong backbone performance, we perform knowledge distillation (KD), using CSPDarknet53 as the teacher and the smaller model as the student. We compare various KD algorithms to identify the technique that produces a smaller model with the modest accuracy drop. According to our experiments, Contrastive Representation Distillation (CRD) yields MobileNetV2 and RepViT with an acceptable accuracy drop. We consider both accuracy drop and model size to choose either MobileNetV2 or RepViT model to replace CSPDarknet53 in the modified YOLOv4 named M-YOLO-CRD and RV-YOLO-CRD. Our evaluation results demonstrate that RV-YOLO-CRD reduces 30% of model size and achieves better mean average precision (mAP) than M-YOLO-CRD. Our experiments show that the M-YOLO-CRD significantly reduces model size (from 245.5 MB to 35.76 MB) and inference time (6× faster on CPU, 4× faster on J. Nano, and 2.5× faster on Orin Nano. While the precision decreased slightly (less than 4%), the model still performs well on edge devices. The M-YOLO-CRD achieved latency per frame at around 37 ms on Orin Nano, 168 ms on J. Nano, and 1310 ms on Raspi4B.

INDEX TERMS Object detection, knowledge distillation, contrastive representation distillation, YOLOv4, MobilenetV2, RepViT, edge computing, model compression.

The associate editor coordinating the review of this manuscript and approving it for publication was Chengpeng Hao.

I. INTRODUCTION

Convolutional Neural Networks (CNNs) have shown substantial progress in object detection (OD) tasks. Prominent

© 2025 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

models such as R-CNN (Region-Based Convolutional Neural Networks) [1], SSD (Single Shot MultiBox Detector) [2], and YOLO (You Only Look Once) [3] have achieved remarkable results. OD models depend significantly on well-crafted features, which are essential for capturing multiple aspects of image structure. Meticulously selected and crafted attributes have resulted in the development of extensive and intricate models. The efficacy of OD models is considerably affected by the backbone architecture that derives high-level features from input photos. An efficient backbone requires significant computational resources.

Edge AI has garnered significant interest due to the rising proliferation of IoT devices and the need for real-time computation in close proximity to sensors [4], [5]. At the edge, OD models can be implemented to execute diverse AI and vision tasks [6]. Nonetheless, due to the restricted computational resources and storage capabilities of industrial IoT and edge devices, OD models' substantial size and complexity have led to significant computing requirements, hindering their implementation on resource-limited edge devices. Model compression techniques [7], [8], [9], [10], [11], [12] have been devised to produce more compact and efficient OD models that tackle this issue. These strategies reduce the parameter count while preserving satisfactory performance.

Contemporary model compression methodologies, including model pruning [7], [8], quantization [9], [10], and knowledge distillation [11], [12], [13], aim to enhance neural network architectures and reduce parameters while preserving satisfactory performance. Specifically, knowledge distillation (KD) is a widely used technology that improves the performance of lightweight networks through an innovative training approach that facilitates knowledge transfer.

KD involves deriving pertinent insights from a large dataset by using a superior teacher model to guide the training of a simpler student model, hence improving its efficacy [14]. Initially utilized for image classification [15], [16], it evolved with various KD-based OD methodologies in 2017 [12], [17]. In KD-based OD, KD is utilized to compress OD models and enhance detection performance in lightweight student models through efficient knowledge transfer [17], [18]. The advent of KD has addressed multiple challenges in various OD tasks. For example, in autonomous driving, KD has been employed to improve models for OD, improving the identification of various objects on the road while maintaining model efficiency. In high-resolution satellite imagery, KD has been applied to improve OD models, aiding in the identification of small objects, such as vehicles and buildings, while preserving accuracy despite model compression.

Contrastive learning has emerged as an efficacious approach for self-supervised tasks [19], [20], [21]. The objective is to reduce the distance between representations of similar inputs while increasing the distance between representations of dissimilar ones, thus encapsulating the relationship between contrasting pairings [22]. Our research incorporates the contrastive learning framework into OD by utilizing encoded information about the interactions among different object regions, building upon these concepts of contrastive learning. Contrastive Representation Distillation (CRD) aims to improve PD performance by extracting knowledge from the contrastive relationships present in the data. Compression and optimization techniques offer two primary advantages. They initially reduce memory usage and computational demands, enabling efficient inference on resource-limited systems. Furthermore, they enable accelerated inference and reduced power consumption, making them suitable for real-time processing applications. This study presents a novel network compression approach for MobileNetV2-based OD, using KD through the M-YOLO-CRD model, as well as for RepViT-based OD, utilizing the RV-YOLO-CRD model, both built on the YOLOv4 architecture.

The assessment of the M-YOLO-CRD model revealed considerable enhancements in model size and inference time compared to the baseline YOLOv4, rendering it exceptionally appropriate for edge devices such as the J. Nano, Orin Nano, and Raspi4B. The model size was reduced sixfold, from 245.53 MB to 35.76 MB. M-YOLO-CRD exhibited an inference speed of 4× faster on the J. Nano, 2.5× faster on the Orin Nano, and 6× faster on the Raspi4B. Although there was a minor decrease in precision (from 81.97% to 78.39%), the performance compromise was warranted due to the significant reduction in computing demand and delay.

Orin Nano achieved a latency of 37.6 ms per frame, enabling real-time processing at around 25 frames per second (FPS), while the J. Nano and Raspi4B exhibited latencies of 167.1 ms and 1310.9 ms per frame, respectively, both representing significant enhancements over the baseline. Furthermore, M-YOLO-CRD exhibited superior power efficiency, especially on the Orin Nano, consuming approximately 1000 mW per frame. The findings validated that M-YOLO-CRD is a proficient and effective solution for real-time OD on resource-limited devices, achieving a balance between compression, speed, and accuracy.

This work has the following scientific contributions:

1) We conducted a comprehensive evaluation to select the acceptable KD from a set of ten KD algorithms, along with four CNN models, to develop a lightweight OD backbone.
2) We propose a modified YOLOv4 model, utilizing CRD on the MobileNetV2 and RepViT backbone for the mini COCO, Car License Plates, and Road Sign datasets.
3) We measure and quantify the performance benefits of our lightweight OD approach, combining YOLOv4, CRD, MobileNetV2 and RepViT on real-world edge devices such as NVIDIA's J. Nano and Orin Nano, as well as Raspi4B.

The remainder of the article is structured as follows. Section II reviews existing methods, followed by an outline of the proposed methods in Section III. Section IV presents the evaluation results and discussions, while the conclusion of this research is summarized in Section V.

II. RELATED WORK

A. OBJECT DETECTION

The advent of deep learning, especially with the creation of CNNs, has markedly progressed the domain of OD. CNNs have exhibited exceptional accuracy and efficiency in image classification and detection tasks, positioning them as the fundamental framework for deep learning-based object recognition.

Two principal methodologies, one-stage and two-stage OD techniques, have been prominent in this field. These approaches are essentially differentiated by their architectural configuration and the number of phases in the detection process. One-stage approaches, such as YOLO [3] and SSD [2], integrate detection and localization into a singular process, frequently employing a dense grid of predetermined bounding boxes. They are optimized for real-time applications and excel in situations requiring swift ODs. Conversely, two-stage approaches, shown by Faster R-CNN [1], function in two separate phases: region proposal generation and object classification/localization. The two-stage method, albeit computationally demanding, provides superior detection accuracy, rendering it appropriate for high-precision jobs such as intricate object recognition and medical imaging.

To meet the computational requirements of intricate OD models on edge devices, numerous model compression approaches have been investigated [6], [23], [24]. The approaches encompass network pruning [7], [8], quantization [9], [10], KD, and optimization of architectural design [25], [26], [27]. Network pruning removes superfluous parameters, whereas quantization diminishes the bit precision of model weights. KD is an efficient method that entails the transfer of knowledge from a bigger, pre-trained model (the teacher model) to a smaller model (the student model).

B. KNOWLEDGE DISTILLATION

Prior research has shown the efficacy of KD in improving OD performance. Chen et al. [12] incorporated KD into Faster R-CNN, enhancing the accuracy-speed balance. Liu et al. [11] utilized KD-based mutual information to improve SSD performance. Bharadhwaj et al. [28] employed ensemble KD to enhance vehicle identification in YOLO Tiny. Kang et al. [29] proposed instance-conditional distillation (ICD), whereas Wang et al. [30] introduced CrossKD, which enables the transfer of intermediate features between teacher and student models. These KD approaches have resulted in substantial enhancements in detection precision and efficacy.

Progress in OD and KD has enabled edge computing and resource-limited edge devices to assume a crucial role in numerous practical applications. An exemplary case is edge-based origin-destination analysis to improve the efficacy and security of intelligent transportation systems. Edge devices may possess sophisticated functionalities like traffic sign detection and license plate identification to facilitate real-time decision-making and diminish dependence on centralized processing. Recent studies have demonstrated the efficacy of KD in enhancing OD models for edge devices. Zhao et al. [31] illustrated the efficacy of KD in lightweight traffic sign identification, whereas Guan et al. [32] combined KD with attention processes in YOLOv5 to enhance performance. Tian et al. [33] presented KDNet, an advanced model for license plate detection. In addition to KD, many compression approaches have been investigated to enhance OD models for edge deployment. Jiang [34] introduced dynamic pruning and weight sharing, whereas Rehman et al. [35] utilized network pruning and training methodologies to improve the identification of minor traffic signs. Wan et al. [36] concentrated on minimizing the model size of YOLOv3.

Although previous studies have shown considerable advancements in OD tasks at the edge, our research possesses distinct originality by concentrating on developing a modified OD by replacing the backbone with lightweight model. To ensure the performance of OD, new backbone has to inherit the knowledge of the original backbone utilizing selected KD algoithms. We also asses modified OD models across various edge devices, encompassing both CPU and GPU platforms.

III. METHODS

YOLOv4 [37], a prevalent OD model, provides superior accuracy. Nonetheless, its substantial size and computational intricacy, chiefly attributed to its CSPDarknet53 backbone, restrict its implementation on resource-limited edge devices. The CSPDarknet53 backbone, originating from CNNs trained on extensive datasets such as ImageNet, enhances the model's size and parameter count. Substituting it with a smaller CNN can considerably decrease the model size, although it frequently results in a significant decline in accuracy for OD activities. Balancing model size and accuracy is an important challenge for developing efficient edge applications.

To tackle this challenge, the smaller backbone must be trained to assimilate essential properties from the larger model. KD provides a viable answer by moving critical knowledge from a substantial teacher model to a more compact student model. Diverse knowledge development approaches have been established, each utilizing a distinct approach to knowledge transfer. For instance, conventional KD emphasizes the transfer of soft labels from the teacher to the learner [13]. FitNet [38] improves this by utilizing intermediate representations, whereas Attention Transfer [39] uses attention maps. Additional methodologies encompass Similarity-Preserving KD [40], Correlation Congruence (CC) [41], Variational Information Distillation (VID) [42], Probabilistic Knowledge Transfer (PKT) [43], Activation Boundaries (AB) [44], Neuron Selectivity Transfer (NST) [45], and CRD [22].

These tactics utilize several strategies to facilitate efficient information transfer and enhance the performance of the smaller student model. For instance, vanilla KD [13] is a

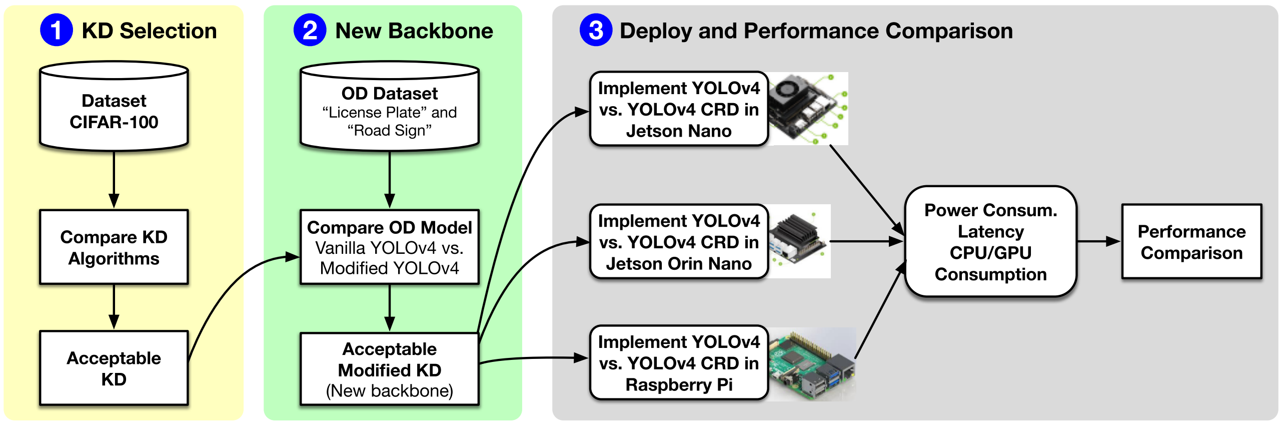
1 KD Selection
Dataset CIFAR-100
Compare KD Algorithms
Acceptable KD
2 New Backbone
OD Dataset "License Plate" and "Road Sign"
Compare OD Model Vanilla YOLOv4 vs. Modified YOLOv4
Acceptable Modified KD (New backbone)
3 Deploy and Performance Comparison
Implement YOLOv4 vs. YOLOv4 CRD in Jetson Nano
Implement YOLOv4 vs. YOLOv4 CRD in Jetson Orin Nano
Implement YOLOv4 vs. YOLOv4 CRD in Raspberry Pi
Power Consum. Latency CPU/GPU Consumption
Performance Comparison

FIGURE 1. **Our research pipeline.**

methodology introduced by Hinton et al. that entails training a smaller model (the student) to replicate the behavior of a bigger, pre-trained model (the teacher). FitNet, developed by Romero et al. [38], enhances the principle of KD by integrating intermediate feature representations into the training procedure. Attention Transfer seeks to synchronize the attention maps of the teacher and student networks. Similarity-preserving KD emphasizes the retention of the data's similarity structure as acquired by the teacher model in the student model. CC for KD seeks to synchronize the correlations between the feature maps of the teacher and student models. Variational Inference Distillation (VID) employs variational inference techniques to convey information from the teacher to the student model. PKT emphasizes transmitting probabilistic information from the teacher to the student model. The distillation of AB emphasizes the transfer of decision boundaries established by the hidden neurons in the teacher model to the student model. KD through Neuron Selectivity Transfer (NST) entails the transference of neuron selectivity patterns from the teacher to the student model.

Given the advancements in KD techniques, we investigate the potential of using KD to compress YOLOv4 while maintaining its detection performance. Instead of simply replacing the backbone, KD offers a more practical approach to transferring critical knowledge from CSPDarknet53 to a smaller backbone.

As shown in Figure 1, our approach follows a systematic procedure with three steps:

1) **KD Selection:** We evaluate the efficacy of multiple KD methods, including vanilla KD, FitNet, Attention Transfer, Similarity-Preserving KD, CC, VID, PKT, AB, NST, and CRD. This comparison seeks to determine the method that reduces accuracy loss between the teacher and student networks.
2) **Building a New Backbone:** Upon identifying the most effective KD approach, we implement it to facilitate knowledge transfer from CSPDarknet53 (teacher) to a more compact backbone (student). The refined smaller backbone is subsequently employed as the replacement backbone for YOLOv4. We assess the efficacy of this modified YOLOv4 model and juxtapose it with the original YOLOv4 and a variant using a smaller backbone without distillation.
3) **Deployment and Performance Comparison:** Upon identifying a model with satisfactory performance, we implement it on edge devices and undertake assessments to evaluate accuracy, latency, and resource utilization. We broaden the comparison to include the improved YOLOv4 model to assess its appropriateness for practical edge device applications.

This study indicates that the original CSPDarknet53 backbone of YOLOv4 functions as the teacher model, while a smaller CNN model is selected as the student due to its diminished size and possible efficiency in edge contexts. Our research entails a comprehensive assessment of the distilled YOLOv4 model on edge devices, demonstrating that KD presents a more efficacious approach than merely substituting the backbone, facilitating high-performance object recognition in resource-limited settings.

TABLE 1. **CIFAR-100, Road Sign, and car license plate dataset.**

Dataset	Number of Classes	Image Size (pixels)	Data Train	Data Test
CIFAR-100 [46]	100	32 × 32	50,000	10,000
Road Sign [47]	4	245 × 400	702	175
Car License Plate [48]	1	400 × 300	347	86
Mini Coco [49]	10	vary	7000	3000

A. DATASET

Three publicly accessible datasets were used in this research, as described below. Table 1 also details these three datasets.

- **CIFAR-100 [46]:** This dataset was used for initial experiments on classifying distillation algorithms, with 50,000 training and 10,000 testing samples. It contains 60,000 color images, each measuring 32 × 32 pixels. With 100 distinct classes, CIFAR-100 is more complex than the original CIFAR-10 dataset.

- **Road Sign Dataset [47]:** This dataset was used for OD tasks with the YOLO-CRD model. It contains 877 images, divided into a training set of 702 images and a testing set of 175 images. The dataset includes four classes: traffic lights, stop signs, speed limits, and crosswalks.
- **Car License Plate Dataset [48]:** This dataset was also used for OD with the YOLO-CRD model. It contains 433 images, divided into a training set of 347 images and a testing set of 86 images.
- **Mini COCO Dataset [49]:** This dataset was also used for various OD tasks with the YOLO-CRD models. The dataset consists of 10,000 images, which are divided into a training set of 7,000 images and a testing set of 3,000 images.

B. KNOWLEDGE DISTILLATION USING CRD AND INTEGRATION INTO YOLOv4

This study presents an OD algorithm employing YOLOv4, incorporating CRD to improve efficiency and precision. YOLOv4 [50] is chosen due to its simplicity and lighter computing requirements than its predecessors, such as YOLOv5 to YOLOv11. YOLOv4 [37] also offers modification flexibility, enabling cutting and replacing the backbone. YOLOv3 and YOLOv4 can achieve a favorable balance between latency and accuracy, comparable to modern YOLO models [50], when improvements such as replacing traditional heads with modern ones and incorporating multiple backbone and neck variations, are applied.

To apply KD to OD models, we first define the loss function that measures the transfer of distilled knowledge from a teacher model to student models, as expressed below.

$$L = L_{\text{det}} + \gamma L_{\text{dis}}, \tag{1}$$

where L_{det} represents the loss function utilized for OD, while L_{dis} denotes the loss employed for KD, with γ serving as a coefficient to balance their respective influences. We examine the methods proposed by integrating KD technologies into traditional two-stage and one-stage OD models to analyze the basic principles of KD-based OD models.

Contrastive learning has emerged as a fundamental element in numerous contemporary investigations of self-supervised representation learning [22]. Techniques like Noise Contrastive Estimation (NCE) assess the similarities across data samples in a deep representation space. These methodologies juxtapose pairs of positive and negative representations, allowing the model to acquire significant distinctions. In KD, CRD enhances this principle by facilitating knowledge transfer from a teacher model to a student model via contrastive loss. CRD enhances the student's capacity to acquire comprehensive and informative representations by contrasting positive and negative pairs within the representation space of both models. This method helps the student capture finer data structure details, effectively resulting in better downstream task performance.

Given an input x_T for the teacher model and x_S for the student model: **Teacher representation** is $z_T = \Phi_T(x_T)$, **Student representation** is $z_S = \Phi_S(x_S)$, where Φ_T and Φ_S are the embedding functions of the teacher and student models, respectively. We use the contrasting loss function in $\mathcal{L}_{\text{CRD}}$, to maximize the similarity between the positive pair (z_T^i, z_S^i) and minimize that between negative pairs $\{(z_T^i, z_S^i) \mid I/i\}$, as defined in equation-(2).

$$\mathcal{L}_{\text{CRD}} = -\log \frac{\exp\left(\frac{\text{sim}(z_T^i, z_S^i)}{T}\right)}{\exp\left(\frac{\text{sim}(z_T^i, z_S^i)}{T}\right) + \sum_{j\neq i} \exp\left(\frac{\text{sim}(z_T^i, z_S^j)}{T}\right)} \tag{2}$$

where $\text{sim}(u, v) = \frac{u^T v}{\|u\| \|v\|}$ represents the similarity between vectors u and v, which is equivalent to the dot product of the L_2 normalized versions of u and v. T is a temperature hyperparameter that scales the logits. Memory bank in CRD stores and manages many feature representations of data samples. The primary purpose is to efficiently supply a vast array of negative samples during training, which is crucial for improving the quality of learned representations in contrastive learning frameworks.

The teacher model is CSPDarkNet53, the principal backbone of YOLOv4, whereas the student model is MobileNetV2 or RepViT, selected for its lightweight architecture and superior performance. The aim is to transmit information from the teacher to the student model, enabling the lightweight MobileNetV2 or RepViT backbone to supplant the old YOLOv4 backbone without degrading performance.

Figure 2 demonstrates that our approach utilizes the compressed backbone for feature extraction. The pre-trained MobileNetV2 and RepViT models were chosen as the backbone of compressed feature extractors, substituting the CSPDarkNet53 network typically used in YOLOv4. We compared the compressed backbones (MobileNetV2 and RepViT). MobileNetV2, a traditional convolution technique for feature extraction, is replaced with depthwise separable convolutions to improve efficiency. In contrast, RepViT replaces traditional convolution processes with a combination of visual transformer methods and re-parameterized convolutions inspired by MobileNetV3, making it well-suited for resource-constrained mobile devices. The distillation process commences with two inputs from the CIFAR100 dataset, supplied to both the teacher and student models. The teacher model employs CSPDarkNet53 as a feature extractor, while the student model utilizes Backbone compressed (MobileNetV2/RepViT). These models the input data to produce feature representations, which are subsequently sent into an embedding space for contrastive learning. Positive and negative sample pairs are generated from these feature representations, with a memory bank retaining negative samples for efficient learning. CRD improves the alignment

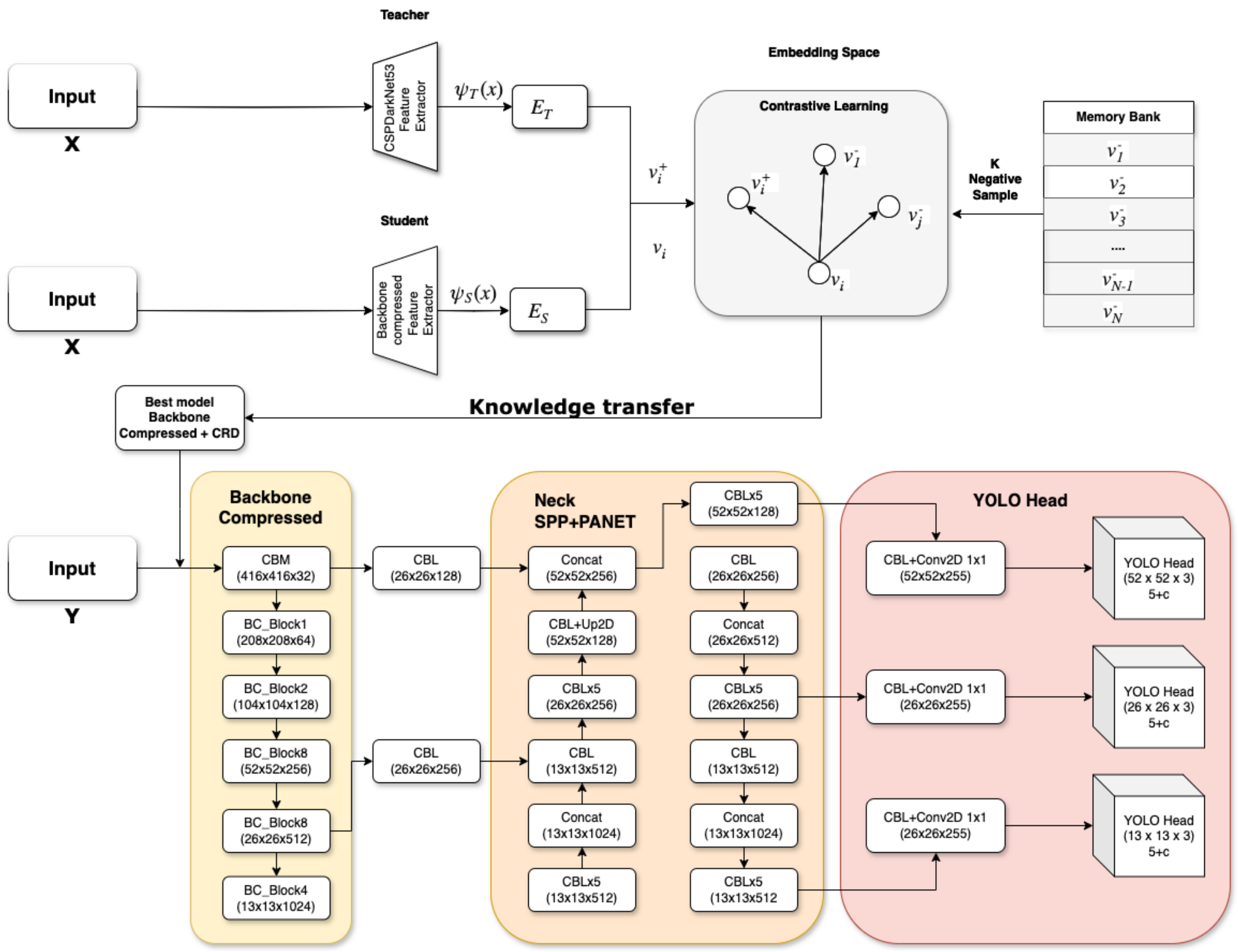

FIGURE 2. **Proposed Method: Compressed backbone with CRD and its integration into YOLOv4.**

of positive pairs while diminishing it for negative pairs, enhancing the student model's efficacy through information transfer from the teacher.

After the distillation process, the best model backbone compressed (MobileNet/RepViT) with CRD is integrated into the YOLOv4 framework as the new backbone. The input images from the Road Sign, Car License Plate, and Mini Coco Dataset [49] are processed through this backbone to extract multi-scale feature maps. These maps are passed into SPPNet to enhance the receptive field and capture multi-scale spatial information, utilizing multiple pooling layers with different kernel sizes (13, 9, and 5). The PANet aggregates these multi-scale feature maps, passing them through a series of Convolutional-BatchNorm-Leaky ReLU (CBL) blocks and up-sampling layers, ensuring robust feature fusion. The final fused feature maps are then passed to the YOLO detection head, predicting bounding boxes, objectness scores, and class probabilities at three scales (52×52, 26×26, and 13×13). We designate the two YOLO-CRD models as M-YOLO-CRD (MobileNetV2 with T = 6, W = 0.5-YOLOv4-CRD) and RV-YOLO-CRD (RepViT 0.6-YOLOv4-CRD). The step-by-step pseudocode of the YOLO-CRD algorithm is explained in Algorithm 1. We also implement the state-of-the-art CrossKD algorithms for entire OD model distillation with the YOLOV4 as the teacher of our proposed YOLOV4 as the students (MobilenetV2-YOLOV4 and RepViT 0.6 - YoloV4) to know the performance compared our proposed methods.

The YOLO-CRD system adeptly integrates the efficiency of MobileNetV2/RepViT with the resilience of CSPDarkNet53, allowing YOLOv4 to sustain its superior performance while minimizing computing requirements. This technique, augmented by SPPNet and PANet, guarantees multi-scale feature extraction and strong OD capabilities, rendering it an appropriate solution for edge devices.

C. EXPERIMENTAL SETUP

1) GPU WORKSTATION

The experiment was conducted on a GPU workstation using a deep learning framework with Python 3.10.12, PyTorch

Algorithm 1 Knowledge Distillation Using Backbone Compressed + CRD and Integration Into YOLOv4

```
1:  Stage 1: Knowledge Distillation Using CRD
2:  Input Processing
3:      Given input image $\mathbf{X}$, up-sample to match the required
        input size for both the teacher and student networks.
4:  Feature Extraction
5:      Teacher Network:
6:          Pass $\mathbf{X}$ through CSPDarkNet53 to obtain feature
            maps $\Phi_T(\mathbf{X})$ and generate teacher embeddings $E_T$.
7:      Student Network:
8:          Pass $\mathbf{X}$ through Compressed Backbone to obtain
            feature maps $\Phi_S(\mathbf{X})$ and generate student embeddings $E_S$.
9:  Contrastive Learning
10:     Form positive pairs $(v_i, v_i^+)$ between student and
        teacher embeddings.
11:     Retrieve negative samples $\{v_j\}$ from the memory bank.
12:     Project embeddings into a shared space for contrastive
        learning.
13: Embedding Space and Memory Bank
14:     Store a large number of negative samples $\{v_j\}$ in the
        memory bank.
15: Knowledge Transfer
16:     Fine-tune backbone compressed using the best weights
        obtained from the CRD process.
17: Stage 2: Integration into YOLOv4 Architecture
18: Backbone compressed + CRD Integration
19:     Integrate the fine-tuned Backbone compressed as the
        new backbone into the YOLOv4 architecture.
20:     Process the input image $\mathbf{y}$ through the backbone to
        extract multi-scale feature maps.
21: SPPNet (Spatial Pyramid Pooling Network)
22:     Feed the extracted feature maps into SPPNet to
        enhance the receptive field.
23: PANet (Path Aggregation Network)
24:     Aggregate feature maps at different scales through
        CBL (Convolutional, BatchNorm, Leaky ReLU)
        blocks and up-sampling.
25: YOLO Head
26:     Predict bounding boxes, objectness scores, and class
        probabilities at three scales.
27: Output
28:     The proposed YOLO-CRD system combines the efficiency of Backbone compressed with the robustness
        of CSPDarkNet53.
```

2.0.1, and CUDA 11.6 to enable GPU resources for OD computations. The hyperparameters for the classification distillation process comprised the SGD optimizer with a momentum of 0.9, an initial learning rate of 0.05, 240 epochs, and a batch size of 64. Various random seeds (11, 37, 42, 50, and 100) were utilized to assess the robustness of the models in OD tasks. Road sign detection utilized 600 epochs, whereas car license detection employed 1200 epochs. The optimizer used was SGD, featuring a momentum value of 0.937.

2) HIGH-PERFORMANCE CLOUD GPU SERVER

We test our proposed model under the Mini COCO dataset [49] for benchmarking purposes. To carry out the test, we use a high-performance cloud computing server with Intel(R) Xeon(R) CPU @ 2.20GHz, A100 GPU, 40 GB GPU memory and 52 GB system memory.

3) EDGE DEVICES

For our evaluation, we used three edge devices: NVIDIA's Orin Nano [51] and J.Nano [52], as well as Raspi4B [53]. Both Orin Nano and Nano have small, yet capable integrated GPUs for accelerated on-device OD inference, while Raspi4B relies on traditional CPUs for OD inferences.

To assess the performance of compressed models in real-world edge implementations, we measured power consumption, latency, memory consumption, CPU consumption, and GPU consumption. Table 2 provides the edge device specifications used in our experiment setup.

TABLE 2. NVIDIA J. Nano, Orin Nano, and Raspi4B specifications.

Specification	J. Nano	Orin Nano	Raspi4B
CPU	4-core ARM Cortex-A57 (@1.43 GHz)	6-core ARM Cortex-A78AE v8.2 (@1.5 GHz)	4-core ARM Cortex-A72 (@1.5 GHz)
GPU	128-core NVIDIA Maxwell GPU	1024-core NVIDIA Ampere GPU with 32 Tensor Cores	-
GPU Max Frequency	921 MHz	625 MHz	-
Memory	4GB LPDDR4	8GB LPDDR5	8GB LPDDR4
Power	5W - 10W	7W - 15W	-
Operating System	Ubuntu 18.04	Ubuntu 20.04	Debian
JetPack	4.6.1	5.1	-

D. EVALUATIONS METRICS

We used accuracy (top-1 accuracy), mAP, and floating-point operations (GFLOPs) as key performance metrics. Additionally, we measured OD inference latency, the utilization of CPU and GPU resources, total power consumption, and power consumption per frame during testing on edge devices. The specific formulas for these metrics are provided in equations-(3) to (7).

Top-1 Accuracy measures how often the model's top predicted class (the one with the highest predicted probability) is the correct class.

$$\text{Top-1 Accuracy} = \frac{\text{Number of correct top predictions}}{\text{Total number of predictions}} \tag{3}$$

Mean Average Precision (mAP) represents the average precision across all categories, indicating the model's overall detection capability. A higher mAP value corresponds to greater precision and accuracy in the model's classifications, as demonstrated below equation-(4).

$$mAP = \frac{1}{m}\sum_{i=1}^{m} AP_i \tag{4}$$

GFLOPs metric measures the total amount of floating-point operations performed during a model's inference phase, measuring computational complexity. This metric is commonly employed to evaluate a model's computational requirements and efficacy, as demonstrated in the equation-(5).

$$\text{GFLOPs} = \frac{\text{FLOPs}}{10^9} \tag{5}$$

Latency denotes the delay or duration required for a data packet to traverse from origin to destination. In edge devices, latency refers to the duration needed for a request to be processed by the device and for a response to be delivered. Low latency is essential for real-time applications, such as video streaming, gaming, and autonomous driving, where rapid response times are imperative, as demonstrated in the equation-(6).

$$\text{Latency} = \frac{\text{Total Inference Time}}{n \text{ Images}}, \tag{6}$$

where 'Total Inference Time' is the cumulative time taken to perform inference on a batch of images. Inference time includes all the steps from loading the data, processing it through the model, and obtaining the output. In this case, n represents the number of images used in the batch to measure the total inference time.

CPU utilization measures CPU consumption during OD operations. CPU utilization is also correlated with the device's energy consumption. High CPU utilization can lead to increased heat generation and energy consumption, potentially impacting the performance of other concurrently running applications if the CPU is heavily utilized.

GPU utilization is to measure the percentage of utilized GPUs. As we utilized two GPU-accelerated edge devices (e.g., J.Nano and Orin Nano) for faster OD inferencing, GPU utilization serves as a critical metric to assess how much of the GPU's processing power is being employed, as it directly impacts latency improvement and power consumption.

Total power consumption refers to the aggregate electrical power a device utilizes during its operational duration. This statistic is generally quantified in watts (W) and includes the total power consumption of the device, comprising the CPU, GPU, memory, and additional components. Effectively regulating total power consumption is essential for edge devices, which frequently function in locations devoid of power grids and depend on portable power banks; thus, power efficiency is paramount.

Power Consumption per Frame (PCF) is measured to quantify the power utilized by the device to process each frame of data during OD operations. Specifically, the power consumption can be often quantified in milliwatts per frame (mW/frame) and offers insight into the device's efficiency in processing visual input. The formula for determining power consumption per frame is expressed in the equation-(7).

$$\text{PCF} = \frac{\text{Total Power Consumption}}{\text{Total Frames Processed}} \tag{7}$$

Lower power consumption per frame indicates a more energy-efficient system, which is advantageous for extending battery life and reducing operational costs in power-constrained environments.

IV. RESULT AND DISCUSSION

A. COMPARISON OF KD METHODS CIFAR 100 IN GPU

Our experiments evaluated the performance of the proposed model considering the metrics discussed in Section III-D. The initial data set used for the classification distillation process is CIFAR100, with 50,000 training samples and 10,000 testing samples. We use CSPDarkNet53 as the teacher and MobileNetV2, RepViT, Resnet14, and ResNet8 as the students.

To evaluate state-of-the-art KD methods, we transferred knowledge from CSPDarknet53 to the student's model for the classification task. As shown in Table 3, CRD distillation consistently outperformed other methods, aligning with previous findings [22]. With a teacher accuracy (CSPDarkNet53) of 68.01%, we experimented with MobileNetV2 (with W = 0.5 and 0.25), RepViT-M0.6, ResNet14 and ResNet8 as student models. Results varied across KD algorithms, demonstrating the importance of knowledge transfer for student model performance.Our goal was to find a student model that effectively captured the knowledge from the teacher. Among students, CRD-MobileNetV2 achieved the highest accuracy with an expansion factor (T=6) and a width multiplier (W=0.5). We also consider RepViT-M0.6 in our experiments to replace CSPDarknet53 in our subsequent YOLOv4 experiments.

B. COMPARISON OF YOLOv4 PERFORMANCE

MobileNetV2 (T=6, W=0.5) with CRD distillation showed the smallest accuracy drop in our previous experiments. We replaced the original CSPDarknet53 backbone in YOLOv4 with MobileNetV2 and evaluated the resulting model. Table 6 compares the size and performance of the baseline YOLOv4 and the compressed model. The M-YOLO-CRD model is six times smaller (64M vs. 9M parameters) and significantly reduces computational complexity. On the other hand, Rv-YOLO-CRD shows light reduction from the 60 M vs. 40 M parameters.

We observe that reducing model parameters can affect the detection performance. Our goal was to achieve a compressed model with minimal precision loss. Tables 4 and 5 present the results of five repetitions of the experiment on the original object detector and various modifications. The average mAP in our five-time repetition was recorded at 81. 97% and 87. 40% on the road sign and the car license plate dataset, respectively.

TABLE 3. Performance Comparison of KD Models in CIFAR-100 [46] based on Teacher: CSPDarkNet53.

Distillation Method \ Student Accuracy (%)	**MobileNetV2 (T=6, W=0.5)**	**MobileNetV2 (T=6, W=0.25)**	**RepViT-M0.6**	**ResNet14**	**ResNet8**
KD Hinton [13]	67.54	58.83	65.21	66.50	57.77
FitNet [38]	65.42	57.43	61.37	66.24	59.93
AT [39]	63.11	51.32	58.26	61.86	53.53
SP [40]	67.74	57.57	9.920	61.25	53.31
CC [41]	64.50	57.30	62.16	66.05	59.90
VID [42]	64.78	56.15	61.86	67.29	59.89
PKT [43]	66.51	58.10	64.51	67.42	58.90
AB [44]	66.90	58.97	n/a	n/a	n/a
NST [45]	64.25	51.00	62.01	61.49	61.71
CRD [22]	**68.33**	**59.35**	**65.71**	**66.34**	**58.67**

T = expansion factor, W = width multiplier

TABLE 4. Comparison of different YOLOv4 vs Rv-YOLO-CRD vs M-YOLO-CRD models with mAP@0.5 on road sign dataset.

OD Model \ mAP@0.5 (%)	**Experiment 1**	**Experiment 2**	**Experiment 3**	**Experiment 4**	**Experiment 5**	**Average ± STD**
YOLOv4-CSPDarkNet53	82.37	81.62	81.62	82.92	81.33	81.97 ± 0.59
YOLOv4-MobileNetV2	76.93	76.39	77.07	77.33	76.06	76.76 ± 0.46
YOLOv4-RepViT	84.69	83.83	81.70	84.71	85.20	84.03 ± 1.24
M-YOLO-CRD	**78.29**	**78.82**	**77.51**	**78.12**	**79.23**	**78.39 ± 0.59**
RV-YOLO-CRD	**82.27**	**82.5**	**83.59**	**82.02**	**82.12**	**82.5 ± 0.57**
M-YOLO-CrossKD	76.61	79.92	72.18	72.23	66.42	73.47 ± 4.57
RV-YOLO-CrossKD	77.57	85.37	82.47	80.41	81.37	81.44 ± 2.55

mAP@0.5 = mean Average Precision at an Intersection over Union (IoU) threshold of 0.5, STD = standard deviation

TABLE 5. Comparison of different YOLOv4 vs Rv-YOLO-CRD vs M-YOLO-CRD models with mAP@0.5 on license plate dataset.

OD Model \ mAP@0.5 (%)	**Experiment 1**	**Experiment 2**	**Experiment 3**	**Experiment 4**	**Experiment 5**	**Average ± STD**
YOLOv4-CSPDarkNet53	88.58	83.42	85.96	89.99	89.03	87.40 ± 2.39
YOLOv4-MobileNetV2	80.31	85.36	84.22	84.29	83.24	83.48 ± 1.72
YOLOv4-RepViT	82.56	82.51	85.29	82.07	83.07	83.10 ± 1.14
M-YOLO-CRD	**86.54**	**85.84**	**88.20**	**86.39**	**83.43**	**86.08 ± 1.54**
RV-YOLO-CRD	**86.66**	**84.03**	**86.31**	**86.31**	**82.36**	**85.13 ± 1.67**
M-YOLO-CrossKD	76.29	78.46	74.87	76.03	79.55	77.04 ± 1.71
RV-YOLO-CrossKD	80.14	74.19	83.29	81.77	82.06	80.29 ± 1.14

mAP@0.5 = mean Average Precision at an Intersection over Union (IoU) threshold of 0.5, STD = standard deviation

TABLE 6. Static Information (Parameters, GFLOPs, and Model Size) of YOLOv4 Models.

OD Model	**Total Parameters**	**Total GFLOPs**	**Model Size**
YOLOv4-CSPDarkNet53	64,363,101	60.527	245.53 MB
M-YOLO-CRD	**9,373,501**	**6.811**	**35.76 MB**
RV-YOLO-CRD	**40,377,973**	**28.789**	**154.03 MB**

To evaluate the robustness of the model, we calculated mAP in five random seeds (11, 37, 42, 50, and 100) in experiments 1 to 5, respectively. We compared baseline YOLOv4, YOLOV4 with backbone replacement, YOLOV4 with backbone replacement, and CRD distillation. We also compared them with the crossKD distillation technique. As shown in Table 6, the modified model is significantly smaller than the original YOLOv4. This reduction in parameters slightly affects object recognition, as evidenced by a drop in mAP from 81.97% to 78.39% on M-YOLO-CRD and 82.5 on Rv-YOLO-CRD. in M-YOLO-CRD, Given the over-six-fold reduction in computational cost, this 2.74% drop is acceptable. On the other hand, Rv-YOLOv-CRD shows better achievement at 82.5%. However, it has a much larger model size compared to M-YOLO-CRD. Our additional cross-KD experiment shows a smaller average mAP of 73. 45% for M-YOLO-CrossKD and 81.44% for RV-YOLO-CrossKD. Both baseline and modified models exhibited similar standard deviations around (0.59%), indicating consistent performance across the five experiments on the road sign dataset. We conducted additional experiments on a license plate dataset to validate consistency, with the results presented in Table 5. Shows the results of our experiment

TABLE 7. Latency Comparison of OD Models on Edge Devices.

Edge Device	Model	Dataset	Latency (ms)
J. Nano	YOLOv4 (CSPDarkNet53)	License Plate	702.972
	YOLOv4 (CSPDarkNet53)	Road Sign	698.704
	M-YOLO-CRD (CRD+MobileNetV2)	License Plate	167.118
	M-YOLO-CRD (CRD+MobileNetV2)	Road Sign	168.452
	RV-YOLO-CRD (CRD+RepViT)	Road Sign	633.22
Orin Nano	YOLOv4 (CSPDarkNet53)	License Plate	97.622
	YOLOv4 (CSPDarkNet53)	Road Sign	97.546
	M-YOLO-CRD (CRD+MobileNetV2)	License Plate	37.634
	M-YOLO-CRD (CRD+MobileNetV2)	Road Sign	37.652
	RV-YOLO-CRD (CRD+RepViT)	Road Sign	89.578
Raspi4B	YOLOv4 (CSPDarkNet53)	License Plate	7523.28
	YOLOv4 (CSPDarkNet53)	Road Sign	7659.54
	M-YOLO-CRD (CRD+MobileNetV2)	License Plate	1310.94
	M-YOLO-CRD (CRD+MobileNetV2)	Road Sign	1317.24
	RV-YOLO-CRD (CRD+RepViT))	Road Sign	n/a

on the car license plate dataset. The result shows similar trends with the previous Table 6. The modified YOLOv4 with CRD achieved a smaller precision drop than the modified YOLOv4 without distillation and the modified YOLOv4 with CrossKD. The M-YOLO-CRD achieved better average precision than the RV-YOLO-CRD at 86. 08% and 85. 13%, respectively, slightly falling from 87. 4% of the original YOLOv4 precision.

The experiment was set five times on various random seeds. The results show that the modified M-YOLO-CRD achieves the second-best precision on average, falling slightly behind the baseline YOLOv4 (CSPDarknet53) despite being 6× smaller and having significantly lower computational costs. Moreover, an evaluation of M-YOLO-CRD and RV-YOLO-CRD on the mini COCO dataset is presented in table 9. This evaluation allows for a fair comparison between our results and state-of-the-art methods. The mean Average Precision (mAP) of M-YOLO-CRD was recorded at 29.36%, slightly below the achievement of RV-YOLO-CRD at 30.46% despite being a quarter model size.

C. COMPRESSED OBJECT DETECTION ON EDGE

A small-size model allowed edge implementation with acceptable memory, CPU, GPU usage, and efficient power consumption. We tested the inference of the proposed model on two types of OD edge device with a graphical processing unit (GPU) and without a GPU. We compare the performance of the original pre-trained baseline YOLOv4, modified YOLO with MobileNetV2 + CRD (M-YOLO-CRD), and modified YOLO with RepViT 0.6+CRD (RV-YOLO-CRD) on target devices. This research uses two edge devices with CPU, J.Nano and Orin Nano, and one without GPU Raspi4B.

Table 7 shows the latency comparison among models on J.Nano, Orin Nano, and Raspi4B. The latency of the compressed YOLOv4 is consistent and faster than the pre-trained baseline. The latency of the M-YOLO-CRD on J.Nano was recorded as 4 times smaller than the baseline model, while on Orin Nano, the gain is a bit lower at about three times faster. On Raspi4B, the latency is one-sixth smaller on a compressed one than the baseline. The smallest latency achieved by Orin Nano is 37.6 milliseconds per frame; in other words, it can be implemented in real-time processing for a video with 25 frames per second. Although the computing efficiency in Raspi4B is six times faster compared to the baseline, it still needs more than one second per frame object recognition on both tested datasets. The latency of RV-YOLO-CRD is slightly smaller than the baseline both in J.Nano and Orin Nano. The smaller speed improvement is caused by the model's size, as shown in table 6. The RV-YOLO-CRD only decreases around 30% from 245 MB to 154 MB. Therefore, it is reasonable if the speed improvement is small.

Figure 3 provided a chart compares the average of GPU usage (%) for two OD tasks: 'license plate detection' and 'road sign detection' on J.Nano and Orin Nano. Two different models are evaluated on each platform: YOLOv4 and M-YOLO-CRD. YOLOv4 is our baseline model with a backbone of CSPDarkNet53, and M-YOLO-CRD is our modified YOLO model with CRD and MobileNetV2. The results confirm that YOLOv4 shows consistently high GPU demands across both platforms, highlighting its computational intensity. In contrast, M-YOLO-CRD is significantly more efficient, particularly on Orin Nano, and we observed that M-YOLO-CRD notably reduces GPU usage, making it a better option for resource-constrained applications while maintaining performance.

Figure 4 shows the average CPU Usage obtained from five trials conducted on the J.Nano, Orin Nano, and Raspi4B with the baseline YOLOv4 and modified M-YOLO-CRD (CRD+MobileNetV2). M-YOLO-CRD consistently uses more CPU resources than the baseline YOLOv4, suggesting that M-YOLO-CRD is more CPU-intensive. Raspi4B

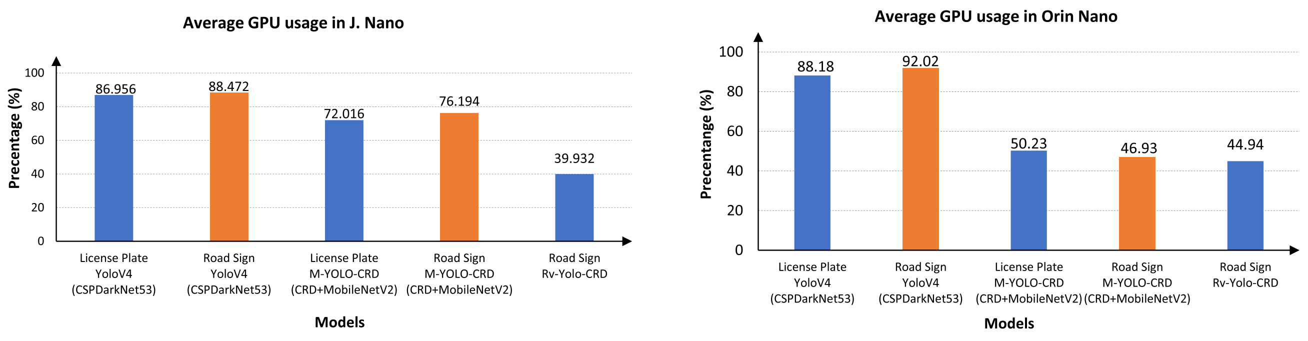

FIGURE 3. Average GPU usage in edge devices based GPU.

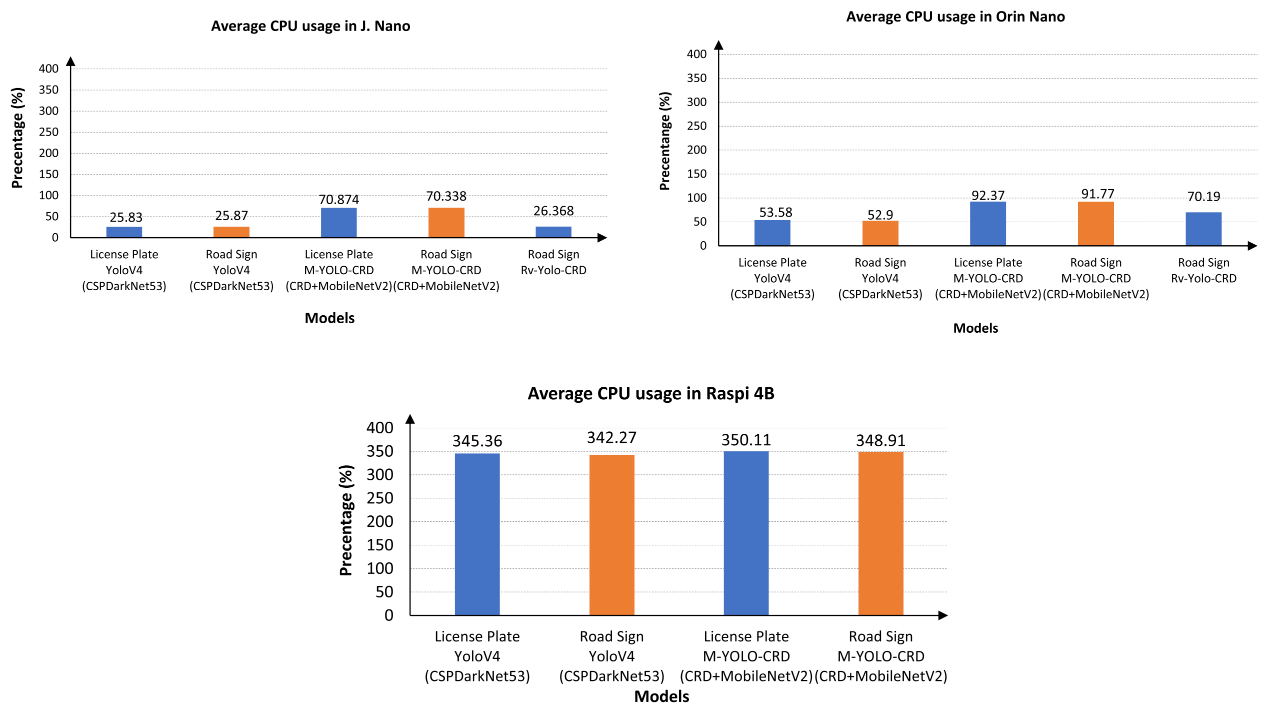

FIGURE 4. Average CPU usage in edge devices.

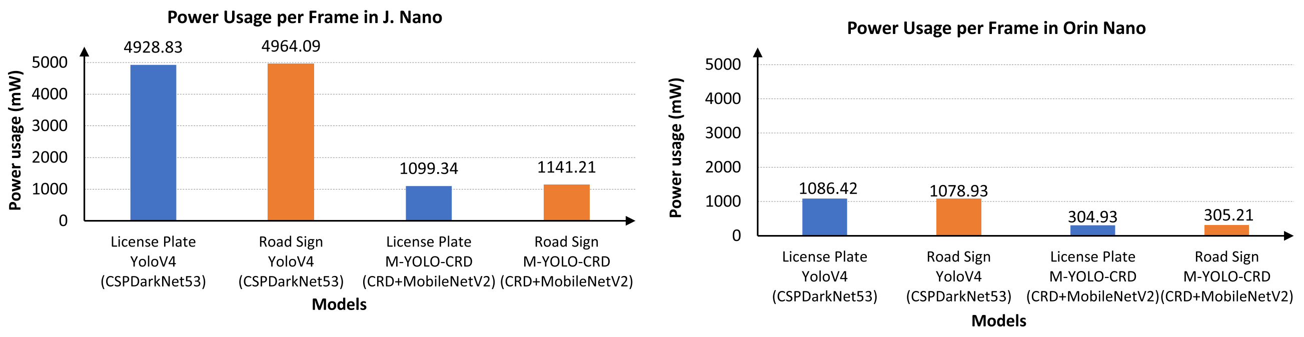

FIGURE 5. Average power usage in edge devices.

struggles with both models, indicating that it may not be ideal for such intensive tasks, as evidenced by the exceedingly high CPU usage percentages, which likely result from the limited processing capabilities of the Raspi4B.

TABLE 8. Comparison with the similar research.

Authors	Device	Model	Evaluation	Results
Archet et al. [54]	J. Nano 8GB	MobileNetV2	Latency energy efficiency	1.13 ms 18.2 GMAC/s/W
Scalcon et al. [55]	J. Nano 8GB	YOLOv4	FPS	13
Pham et al. [56]	Orin Nano 8GB	ResNet50	RAM usage FPS	3.11 Gb 47.56
Wang et al. [57]	MAX78000 128KB	DNN	Power Inference time Energy	20.08 mW 91.9 ms 1.845 mJ

TABLE 9. Comparison in Mini Coco [49] Dataset.

Source	Method	Backbone	mAP@0.5	Parameters(M)	GFLOPs
Yifan Shao [58]	YoloV8	MobileNetV3	14.6	1.42	4.0
Yifan Shao [58]	YoloV8	MobileNetV3+MHSA	11.7	1.45	4.0
Yifan Shao [58]	YoloV8	MobileNetV3+SE	15.0	1.45	4.0
Yifan Shao [58]	YoloV8	MobileNetV3+CBAM	13.8	1.47	4.0
Yifan Shao [58]	YoloV8	MobileNetV3+LGA	15.1	1.42	4.0
M-YOLO-CRD (ours)	YOLOv4	MobileNetV2	29.36	9.3	6.8
RV-YOLO-CRD (ours)	YOLOv4	RepViT	30.46	40.3	28.7

Figure 5 shows that the J. Nano consumes significantly more power, both in total and per frame, when using the YOLOv4 models compared to the more efficient M-YOLO-CRD models, with power consumption of approximately 1100 mW per frame on J.Nano. Orin Nano also shows a similar trend, but with a lower total power usage and power usage per frame compared to J. Nano, approximately 300 mW per frame, making it more power-efficient. M-YOLO-CRD models are clearly more energy-efficient across both devices, demonstrating significantly reduced power consumption compared to YOLOv4 models.

D. COMPARISON WITH THE SIMILAR RESEARCH

When M-YOLO-CRD and RV-YOLO-CRD are compared with other models in similar research, several key differences in performance and efficiency become apparent. With regard to latency, M-YOLO-CRD significantly outperforms many models, especially on edge devices. For example, on the J. Nano, M-YOLO-CRD achieves a latency of 167.1 ms for license plate detection, which is approximately four times faster than the baseline YOLOv4, which takes 702.97 ms as shown in Table 8. In comparison, Archet et al.'s [54] MobileNetV2 achieved a much lower latency of 1.13 ms, but this was for a more straightforward image classification task rather than the more computationally intensive OD. Wang et al.'s MAX78000-based DNN, on the other hand, had a slower inference time of 91.9 ms, making M-YOLO-CRD better suited for real-time OD on edge devices.

To enable a fair comparison with other works, an evaluation on the mini-COCO dataset is conducted. Table 9 compares M-YOLO-CRD and RV-YOLO-CRD in terms of precision, parameter count, and GFLOPS. M-YOLO-CRD achieves a precision of 29.36, outperforming all state-of-the-art models built on newer versions of YOLO. However, its precision is slightly below that of RV-YOLO-CRD. This higher precision comes at the cost of increased computational complexity, as M-YOLO-CRD requires 6.8 GFLOPS compared to 4.0 for its closest competitor. Despite this, M-YOLO-CRD nearly doubles the precision, improving from 15% to 29%. In contrast, while RV-YOLO-CRD achieves the highest precision, its computational cost is significantly higher, with GFLOPS recorded at 28.7, which is approximately 7× heavier than M-YOLO-CRD. This trade-off highlights the balance between precision and computational efficiency across the evaluated models.

Regarding energy efficiency, MobileNetV2 [54] achieved an impressive 18.2 GMAC/s/W on the J. Nano platform, but this result pertains to an image classification task. OD, which requires greater computational power, typically consumes more energy. M-YOLO-CRD was designed to address this by reducing both computational and energy costs compared to the baseline YOLOv4 with CSPDarkNet53. It achieves this by being six times smaller in terms of parameters while significantly lowering GPU and power consumption. Although Wang et al. [57] demonstrated extremely low power consumption (20.08 mW) and energy usage (1.845 mJ) with the MAX78000, their model was much slower, making it more suitable for low-power tasks rather than real-time OD. In contrast, M-YOLO-CRD strikes a balance between lower energy consumption and maintaining detection speed and accuracy, making it well-suited for edge applications requiring moderate power use.

When examining FPS, Pham et al.'s [56] ResNet50 achieved a high 47.56 FPS on the Orin Nano, optimized for high-speed applications like video processing. However, Scalcon et al. [55] YOLOv4 on the J. Nano managed 13 FPS, a trade-off for accuracy in OD tasks. Although M-YOLO-CRD does not have a specific FPS value reported, its reduced latency and real-time performance suggest that it performs competitively, especially given its reduced computational complexity.

Regarding memory usage and model size, Pham et al.'s ResNet50 consumed 3.11 GB of RAM, demonstrating efficiency for a model of its complexity on the Orin Nano. In comparison, M-YOLO-CRD excels in this area by significantly reducing model size, with only 9.37 million parameters compared to 64.36 million parameters in the baseline YOLOv4. This reduction makes M-YOLO-CRD much more memory-efficient and better suited for deployment on edge devices with limited resources, all while maintaining reasonable detection accuracy.

Overall, M-YOLO-CRD offered a balanced option by combining low latency, reduced power consumption, and a smaller model size, making it highly suitable for real-time OD tasks on edge devices. While models, e.g., those proposed by Archet et al. [54] and Wang et al. [57], excel in specific aspects like latency or power efficiency, M-YOLO-CRD demonstrated versatility by delivering significant reductions in computational costs with minimal impact on accuracy or real-time performance. This makes it an excellent choice for applications requiring compact, energy-efficient models with minimal accuracy trade-offs.

V. CONCLUSION

This research focused on compressing OD models by replacing the backbone using smaller network. We presented CRD successfully transferring knowledge from the CSPDarknet53 as a teacher to smaller networks like MobileNetV2, ResNet14, ResNet8, and RepViT-M0.6. Compared to other KD algorithms, the student models trained by CRD exhibited smaller accuracy losses. Our proposed method named M-YOLO-CRD replaced CSPDarknet53 with MobileNetV2 in YOLOv4 and significantly reduced model size, number of parameters, and computational complexity by six times while maintaining detection performance with minimal average precision loss (less than 4%). This enabled the deployment of the OD model on edge devices.

Additionally, a comparison with a modified YOLOv4 incorporating the RepViT backbone is presented. When evaluating model compression and precision loss, YOLOv4 with RepViT-M0.6 backbone and CRD distillation (Rv-YOLO-CRD) achieved slightly better precision than M-YOLO-CRD on the mini-COCO and Road Sign datasets. However, Rv-YOLO-CRD, which is only about 30% smaller than the baseline or approximately 4× larger than M-YOLO-CRD, suffers from high latency.

We evaluate the M-YOLO-CRD on real-world edge devices, and the evaluation results were promising. Latency was reduced by 2.5× on the Orin Nano, 4× on the J.Nano, and 6× on the Raspi4B. Orin Nano demonstrated the best latency, achieving 37.6 ms per frame, while J.Nano and Raspi4B exhibited slower latencies of slightly over 160 ms and 1300 ms, respectively. On GPU-equipped edge devices (Orin Nano and J. Nano), the workload was distributed between the CPU and GPU, resulting in lower latency compared to the Raspi4B, which lacks GPU accelerators. We observed that GPU utilization of the baseline YOLOv4 was 100%, whereas the compressed model reduced GPU utilization to 70% on the J.Nano and 50% on Orin Nano. CPU utilization was relatively low on Orin Nano and J. Nano; however, we also observed that Raspi4B utilized four cores at over 300%. Given the importance of power consumption for edge devices, we also measured inference power consumption, which was approximately 1100 mW per frame on the J.Nano and 300 mW on the Orin Nano.

As a future direction, we aim to explore compressing the entire OD model, as demonstrated in newer YOLO versions (YOLOv5 to YOLOv10). These versions offer ''fat'' and ''tiny'' variants, with the tiny versions prioritizing inference speed and resource efficiency by reducing the number of parameters. Additionally, investigating the transfer of knowledge from large teacher models to tiny OD models using KD could lead to even more compact and efficient models.

ACKNOWLEDGMENT

The authors are grateful to the Research Directorate of Universitas Amikom Yogyakarta for the support provided to this project.

REFERENCES

[1] R. Gavrilescu, C. Zet, C. Foşalău, M. Skoczylas, and D. Cotovanu, "Faster R-CNN: An approach to real-time object detection," in *Proc. Int. Conf. Expo. Electr. Power Eng. (EPE)*, Oct. 2018, pp. 165–168.

[2] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Jan. 2016, pp. 21–37.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[4] J. Hao, P. Subedi, L. Ramaswamy, and I. K. Kim, "Reaching for the sky: Maximizing deep learning inference throughput on edge devices with AI multi-tenancy," *ACM Trans. Internet Technol.*, vol. 23, no. 1, pp. 1–33, Feb. 2023.

[5] P. Subedi, J. Hao, I. K. Kim, and L. Ramaswamy, "AI multi-tenancy on edge: Concurrent deep learning model executions and dynamic model placements on edge devices," in *Proc. IEEE Int. Conf. Cloud Comput. (CLOUD)*, 2021, pp. 31–42.

[6] A. Setyanto, T. B. Sasongko, M. A. Fikri, and I. K. Kim, "Near-edge computing aware object detection: A review," *IEEE Access*, vol. 12, pp. 2989–3011, 2024.

[7] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peşte, "Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks," *J. Mach. Learn. Res.*, vol. 22, no. 1, pp. 1–124, 2021.

[8] S. Ghosh, S. K. K. Srinivasa, P. Amon, A. Hutter, and A. Kaup, "Deep network pruning for object detection," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2019, pp. 3915–3919.

[9] R. Li, Y. Wang, F. Liang, H. Qin, J. Yan, and R. Fan, "Fully quantized network for object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2805–2814.

[10] P. Chen, J. Liu, B. Zhuang, M. Tan, and C. Shen, "AQD: Towards accurate quantized object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 104–113.

[11] X. Liu and Z. Zhu, "Knowledge distillation for object detection based on mutual information," in *Proc. 4th Int. Conf. Intell. Auto. Syst. (ICoIAS)*, May 2021, pp. 18–23.

[12] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, "Learning efficient object detection models with knowledge distillation," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 1–10.

[13] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.

[14] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *Int. J. Comput. Vis.*, vol. 129, no. 6, pp. 1789–1819, Jun. 2021.

[15] J. Gou, L. Sun, B. Yu, S. Wan, and D. Tao, "Hierarchical multi-attention transfer for knowledge distillation," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 20, no. 2, pp. 1–20, Feb. 2024.

[16] D. Walawalkar, Z. Shen, and M. Savvides, "Online ensemble model compression using knowledge distillation," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 18–35.

[17] S. S. S. Kruthiventi, P. Sahay, and R. Biswal, "Low-light pedestrian detection from RGB images using multi-modal knowledge distillation," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 4207–4211.

[18] R. Yu, A. Li, V. I. Morariu, and L. S. Davis, "Visual relationship detection with internal and external linguistic knowledge distillation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1068–1076.

[19] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2018, *arXiv:1807.03748*.

[20] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 9726–9735.

[21] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. 37th Int. Conf. Mach. Learn. (ICML)*, 2020, pp. 1597–1607.

[22] Y. Tian, D. Krishnan, and P. Isola, "Contrastive representation distillation," 2020, *arXiv:1910.10699*.

[23] R. D. Rachmanto, A. N. L. Nabhaan, and A. Setyanto, "Deep learning model compression techniques performance on edge devices," *MATRIK, Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*, vol. 23, no. 3, pp. 569–582, Jun. 2024.

[24] R. D. Rachmanto, Z. Sukma, A. N. L. Nabhaan, A. Setyanto, T. Jiang, and I. K. Kim, "Characterizing deep learning model compression with post-training quantization on accelerated edge devices," in *Proc. IEEE Int. Conf. Edge Comput. Commun. (EDGE)*, Jul. 2024, pp. 110–120.

[25] Y. Yin, H. Li, and W. Fu, "Faster-YOLO: An accurate and faster object detection method," *Digit. Signal Process.*, vol. 102, Jul. 2020, Art. no. 102756.

[26] R. J. Wang, X. Li, and C. X. Ling, "Pelee: A real-time object detection system on mobile devices," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2018, pp. 1967–1976.

[27] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10778–10787.

[28] M. Bharadhwaj, G. Ramadurai, and B. Ravindran, "Detecting vehicles on the edge: Knowledge distillation to improve performance in heterogeneous road traffic," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2022, pp. 3191–3197.

[29] Z. Kang, P. Zhang, X. Zhang, J. Sun, and N. Zheng, "Instance-conditional knowledge distillation for object detection," in *Proc. 35th Int. Conf. Neural Inf. Process. Syst* Red Hook, NY, USA: Curran Associates, 2021, pp. 16468–16480.

[30] J. Wang, Y. Chen, Z. Zheng, X. Li, M.-M. Cheng, and Q. Hou, "CrossKD: Cross-head knowledge distillation for object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2024, pp. 16520–16530.

[31] L. Zhao, Z. Wei, Y. Li, J. Jin, and X. Li, "SEDG-YOLOv5: A lightweight traffic sign detection model based on knowledge distillation," *Electronics*, vol. 12, no. 2, p. 305, Jan. 2023.

[32] P.-W. Guan and W.-X. Zhu, "Knowledge distillation and attention mechanism analysis of traffic sign detection," in *Proc. China Autom. Congr. (CAC)*, Nov. 2022, pp. 2686–2691.

[33] C. Tian, X. Zhang, X. Liang, B. Li, Y. Sun, and S. Zhang, "Knowledge distillation with fast CNN for license plate detection," *IEEE Trans. Intell. Vehicles*, early access, Nov. 6, 2023, doi: 10.1109/TIV.2023.3330164.

[34] Q. Jiang, T. Rui, J. Dai, F. Shao, G. Lu, and J. Wang, "A real-time detection method of multi-scale traffic signs based on dynamic pruning strategy," *Multimedia Tools Appl.*, vol. 82, no. 21, pp. 32519–32537, Sep. 2023.

[35] Y. Rehman, H. Amanullah, M. A. Shirazi, and M. Y. Kim, "Small traffic sign detection in big images: Searching needle in a hay," *IEEE Access*, vol. 10, pp. 18667–18680, 2022.

[36] J. Wan, W. Ding, H. Zhu, M. Xia, Z. Huang, L. Tian, Y. Zhu, and H. Wang, "An efficient small traffic sign detection method based on YOLOv3," *J. Signal Process. Syst.*, vol. 93, no. 8, pp. 899–911, Aug. 2021.

[37] M. Ning, Y. Lu, W. Hou, and M. Matskin, "YOLOv4-object: An efficient model and method for object discovery," in *Proc. IEEE 45th Annu. Comput., Softw., Appl. Conf. (COMPSAC)*, Jul. 2021, pp. 31–36.

[38] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "FitNets: Hints for thin deep nets," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–13.

[39] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–13.

[40] F. Tung and G. Mori, "Similarity-preserving knowledge distillation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1365–1374.

[41] B. Peng, J. Xiao, D. Li, S. Zhou, Y. Wu, J. Liu, Z. Zhang, and Y. Liu, "Correlation congruence for knowledge distillation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 5006–5015.

[42] S. Ahn, S. X. Hu, A. Damianou, N. D. Lawrence, and Z. Dai, "Variational information distillation for knowledge transfer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9155–9163.

[43] N. Passalis and A. Tefas, "Learning deep representations with probabilistic knowledge transfer," in *Proc. 15th Eur. Conf. Comput. Vis. (ECCV)*, Jan. 2018, pp. 283–299.

[44] B. Heo, M. Lee, S. Yun, and J. Y. Choi, "Knowledge transfer via distillation of activation boundaries formed by hidden neurons," in *Proc. 33rd AAAI Conf. Artif. Intell. (AAAI)*, vol. 33, Jul. 2019, pp. 3779–3787.

[45] Z. Huang and N. Wang, "Like what you like: Knowledge distill via neuron selectivity transfer," 2017, *arXiv:1707.01219*.

[46] A. Krizhevsky, V. Nair, and G. Hinton. *CIFAR-100 (Canadian Institute for Advanced Research)*. [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html

[47] *Road Signs Dataset*. [Online]. Available: https://makeml.app/datasets/road-signs

[48] *Car License Plates Dataset*. Accessed: Feb. 13, 2024. [Online]. Available: https://www.kaggle.com/datasets/andrewmvd/car-plate-detection

[49] N. Samet, S. Hiçsönmez, and E. Akbaş, "HoughNet: Integrating near and long-range evidence for bottom-up object detection," in *Proc. 16th Eur. Conf.*, Glasgow, U.K. Springer, Aug. 2020, pp. 406–423.

[50] I. Lazarevich, M. Grimaldi, R. Kumar, S. Mitra, S. Khan, and S. Sah, "YOLOBench: Benchmarking efficient object detectors on embedded systems," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2023, pp. 1161–1170. [Online]. Available: https://api.semanticscholar.org/CorpusID

[51] (2024). *Jetson Orin*. Accessed: Jul. 16, 2024. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin

[52] (2024). *Jetson Nano*. Accessed: Jul. 16, 2024. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano

[53] W. Gay, *Raspberry PI Hardware Reference*, 1st ed., Apress, 2014. Accessed: Jul. 16, 2024.

[54] A. Archet, N. Gac, F. Orieux, and N. Ventroux, "Embedded AI performances of Nvidia's Jetson Orin SoC series," in *Proc. 17th Nat. Colloq. GDR SOC2*, 2023, pp. 1–3.

[55] F. P. Scalcon, R. Tahal, M. Ahrabi, Y. Huangfu, R. Ahmed, B. Nahid-Mobarakeh, S. Shirani, C. Vidal, and A. Emadi, "AI-powered video monitoring: Assessing the NVIDIA Jetson orin devices for edge computing applications," in *Proc. IEEE Transp. Electrific. Conf. Expo (ITEC)*, Jun. 2024, pp. 1–6.

[56] H. V. Pham, T. G. Tran, C. D. Le, A. D. Le, and H. B. Vo, "Benchmarking Jetson edge devices with an end-to-end video-based anomaly detection system," in *Proc. Future Inf. Commun. Conf.*, in Lecture Notes in Networks and Systems, vol. 920, 2024, pp. 358–374.

[57] G. Wang, Z. P. Bhat, Z. Jiang, Y. W. Chen, D. Zha, A. C. Reyes, A. Niktash, G. Ulkar, E. Okman, X. Cai, and X. Hu, "Bed: A real-time object detection system for edge devices," in *Proc. 31st ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2022, pp. 4994–4998.

[58] Y. Shao, "Local–global attention: An adaptive mechanism for multi-scale feature integration," 2024, *arXiv:2411.09604*.

ARIEF SETYANTO (Member, IEEE) received the Ph.D. degree in electronics and computer science from the University of Essex, Colchester, CO, U.K., in 2016. Currently, he is an Associate Professor with the Department of Informatics, Universitas Amikom Yogyakarta, Indonesia. His research interests include image segmentation, image classification, object detection, near-edge computing, and deep learning.

THEOPILUS BAYU SASONGKO received the bachelor's degree in information systems from Universitas Kristen Duta Wacana, Yogyakarta, in 2012, and the master's degree in electrical engineering from Gadjah Mada University, in 2014. In 2017, he joined Universitas Amikom Yogykarta, Indonesia, as a Lecturer. His research interests include data mining, artificial intelligence, deep learning, and computer vision.

MUHAMMAD AINUL FIKRI received the bachelor's degree in telecommunication and audio engineering from Andalas University, Indonesia, and the master's degree in artificial intelligence and human–computer interaction from Gadjah Mada University. He received a scholarship from the Education Fund Management Agency (LPDP) under the Ministry of Finance of the Republic of Indonesia to received the master's degree.

DHANI ARIATMANTO received the master's degree in informatics from Universiti Amikom Yogyakarta, Indonesia, and the Ph.D. degree in image processing from the Faculty of Computing, Universiti Malaysia Pahang, in 2021. He is currently a Lecturer with the Faculty of Computer, Universitas Amikom Yogyakarta. His research interests include image processing, artificial intelligence, multimedia applications, software engineering, and computer vision.

I. MADE ARTHA AGASTYA received the bachelor's degree in aeronautical engineering from Bandung Institute of Technology, Indonesia, the master's degree in electrical engineering from Gadjah Mada University, Yogyakarta, Indonesia, and the Ph.D. degree in computer science from Taylor's University, Malaysia. He is currently an Informatics Lecturer at Universitas Amikom Yogyakarta, Indonesia. His research interests include electroencephalogram (EEG), brain–computer interface (BCI), computer vision, and deep learning.

RAKANDHIYA DAANII RACHMANTO received the bachelor's degree from Universitas Amikom Yogyakarta, in 2024. He is currently pursuing the Ph.D. degree with the University of Georgia, Athens, GA, USA. He received the Garuda Academic and Research of Excellence (Garuda ACE) funded by Indonesia Endowment Fund for Education Agency (LPDP), from 2022 to 2023, and the Ministry of Education Republic Indonesia. His research interests include model compression and edge computing.

AFFAN ARDANA received the bachelor's degree in informatics from Universitas Amikom Yogyakarta, in 2024. Currently, he is a Research Assistant with the Object Detection Research Group funded by the Ministry of Education Republic Indonesia. His research interests include deep learning compression, edge computing, and computer vision.

IN KEE KIM (Member, IEEE) received the Ph.D. degree in computer science from the University of Virginia, Charlottesville, VA, USA, in 2018. He is currently an Associate Professor with the School of Computing, University of Georgia. His current projects specifically focus on edge AI, server-less workflow management, and reproducible benchmarking. His research interests include distributed systems, cloud computing, and edge computing, with a focus on system- and application-level optimizations to enhance scalability, performance, and cost efficiency. He is a member of ACM.