https://doi.org/10.31436/iiumej.v27i3.4344

# Efficient Edge-Based Object Detection via Multi-Signal Teacher–Student Knowledge Distillation

**RICHA SHARMA[1,2], ASHUTOSH GUPTA[1], PRASUN CHAKRABARTI[1], ARVIND SHARMA[1]**

*[1]Faculty of Computing and Informatics, Sir Padampat Singhania University, Udaipur, India*
*[2]Department of Information Technology, Dwarkadas J. Sanghvi College of Engineering, Mumbai, India*

*Corresponding author: richa.gairola@gmail.com*

*(Received: 16 April 2026; Accepted: 11 August 2026; Published online: 11 September 2026)*

***ABSTRACT*:** A key obstacle to bringing computer vision everywhere, from robots and drones to smart cities and wearable tech, is the difficulty of deploying real-time object detectors on resource-limited edge hardware such as the NVIDIA Jetson Nano, Raspberry Pi 5, and ARM-based mobile processors. State-of-the-art detectors such as YOLOv8-L achieve outstanding accuracy on server GPUs but often deliver only single-digit or low FPS on embedded hardware. This paper presents EdgeKD-Net, a multi-signal Knowledge Distillation (KD) framework that transfers the representational knowledge of a high-capacity teacher to a compact edge-deployable student. The student employs a MobileNetV3-Large backbone augmented with a quantization- and distillation-aware Dual-Path Feature Pyramid (DPFP). Three complementary distillation signals are introduced: (i) temperature-scaled response-level KL divergence ($\tau$=4); (ii) Channel-Affinity Feature Alignment (CAFA), transferring structural channel co-activation patterns via cosine-affinity matrices; and (iii) Detection Fidelity Loss (DFL), applied exclusively on the formally defined consistent anchor set, eliminating noisy gradients from anchor-assignment mismatch. Across three independent runs, EdgeKD-Net achieves a mAP of 38.6 ± 0.14 mAP@[0.50:0.95] on MS-COCO 2017, only 1.4 mAP below its YOLOv8-L teacher while operating at 61 FPS (FP16) and 89 FPS (INT8) on NVIDIA Jetson Nano with 8.77 mAP/W energy efficiency. The framework is also evaluated on the VisDrone 2023 dataset, which contains dense aerial small objects. It achieves 14.1 ± 0.22 mAP@[0.50:0.95], compared with 12.4 for the best distillation baseline. This improvement is larger than the 0.9-point gain on MS-COCO, showing that the proposed method is effective on both datasets. The evaluation is limited to these two datasets, and we do not claim broader generalization.

***ABSTRAK*:** Halangan utama dalam memperluas penggunaan penglihatan komputer, daripada robot dan dron kepada bandar pintar serta teknologi boleh pakai, ialah kesukaran melaksanakan pengesan objek masa nyata pada perkakasan pengkomputeran pinggir yang mempunyai sumber terhad, seperti NVIDIA Jetson Nano, Raspberry Pi 5 dan pemproses mudah alih berasaskan ARM. Pengesan tercanggih seperti YOLOv8-L mencapai ketepatan yang sangat baik pada unit pemprosesan grafik (GPU) pelayan, tetapi lazimnya hanya menghasilkan kadar bingkai sesaat (FPS) satu digit atau rendah apabila dilaksanakan pada perkakasan terbenam. Makalah ini memperkenalkan EdgeKD-Net, iaitu kerangka Penyulingan Pengetahuan (Knowledge Distillation, KD) berbilang isyarat yang memindahkan pengetahuan representasi daripada model pengajar berkapasiti tinggi kepada model pelajar kompak yang sesuai dilaksanakan pada peranti pinggir. Model pelajar menggunakan tulang belakang MobileNetV3-Large yang dipertingkatkan dengan Piramid Ciri Dwialiran (Dual-Path Feature Pyramid, DPFP) yang peka terhadap pengkuantuman dan penyulingan. Tiga isyarat penyulingan yang saling melengkapi diperkenalkan: (i) pencapahan Kullback–Leibler (KL) pada aras respons dengan penskalaan suhu ($\tau$ = 4); (ii) Penjajaran Ciri

Afiniti Saluran (Channel-Affinity Feature Alignment, CAFA), yang memindahkan pola struktur pengaktifan bersama saluran melalui matriks afiniti kosinus; dan (iii) Kehilangan Kesetiaan Pengesanan (Detection Fidelity Loss, DFL), yang digunakan secara eksklusif pada set sauh tekal yang ditakrifkan secara formal bagi menghapuskan kecerunan hingar akibat ketidakpadanan penetapan sauh. Berdasarkan tiga larian bebas, EdgeKD-Net mencapai mAP sebanyak 38.6 ± 0.14 bagi mAP@[0.50:0.95] pada set data MS-COCO 2017, iaitu hanya 1.4 mata mAP lebih rendah daripada model pengajarnya, YOLOv8-L. Pada NVIDIA Jetson Nano, model ini beroperasi pada 61 FPS menggunakan FP16 dan 89 FPS menggunakan INT8, dengan kecekapan tenaga sebanyak 8.77 mAP/W. Kerangka ini turut dinilai menggunakan set data VisDrone 2023 yang mengandungi objek udara bersaiz kecil dan padat. EdgeKD-Net mencapai 14.1 ± 0.22 mAP@[0.50:0.95], berbanding 12.4 yang dicatatkan oleh kaedah asas penyulingan terbaik. Peningkatan ini lebih besar daripada peningkatan 0.9 mata pada MS-COCO, sekali gus menunjukkan bahawa kaedah yang dicadangkan berkesan pada kedua-dua set data. Walau bagaimanapun, penilaian kajian ini terhad kepada kedua-dua set data tersebut dan tiada dakwaan dibuat mengenai keupayaan generalisasi yang lebih luas.

***KEYWORDS***: *Knowledge distillation, object detection, edge computing, anchor-assignment consistency, FPN, aerial object detection*

## 1. INTRODUCTION

Deploying vision systems on robots, drones, and embedded cameras is, in practice, more challenging than it appears. These devices need to detect objects in real time, typically at 20–30 frames per second or faster, while drawing only a few watts of power. Models like YOLOv8 [1], YOLOv9 [30], RT-DETR [2], and DINO [3] hit impressive accuracy numbers on server benchmarks, but their inference speed collapses on embedded hardware. A YOLOv8-L running on an NVIDIA Jetson Nano, for instance, manages around 7 FPS. This falls well short of the requirements of a delivery robot that needs a new perception update every 50 ms.

Knowledge distillation (KD) [4] is one of the more practical approaches to bridging this gap. The central idea is to train a lightweight student model to replicate the behavior of a heavier teacher, transferring learned knowledge without carrying over the computational cost. For image classification, this performs well and has been studied extensively [5-7]. Detection is a harder case, however, and this work identifies a specific issue that, to our knowledge, has not been formally addressed before.

The problem is anchor-assignment mismatch. In anchor-based detectors, each spatial position in the feature map is associated with one or more anchor boxes, and each anchor is assigned responsibility for detecting a particular object. When the teacher and student use different backbones and therefore develop different spatial feature representations, they often disagree on which anchor should handle which object. Existing knowledge distillation methods, including FGD [11], MGD [12], DeFeat [13], and PKD [9], apply distillation uniformly across all anchors, regardless of whether the teacher and student agree at that position. When they disagree, the student receives conflicting supervisory signals. Our experiments show this happens at roughly 22.9% of matched anchor positions across the COCO training set, and that naively distilling through those positions costs approximately 0.5 mAP.

We built EdgeKD-Net to address this, along with two other issues: the capacity gap between teacher and student feature representations (handled by CAFA) and the student neck architecture (handled by DPFP). We settled on MobileNetV3-Large [14] as the backbone after running latency measurements on an NVIDIA Jetson Nano at 416×416 resolution. It delivered 44 FPS at 4.0 W, which beat both EfficientViT-M0 (37 FPS / 4.8 W) and FastViT-T8 (41 FPS

/ 4.3 W) under the same conditions. The complete framework, EdgeKD-Net, makes the following contributions:

- A quantization- and distillation-aware dual-path pyramid (DPFP) that integrates top-down semantic context with bottom-up spatial detail at a cost of 1.8 M additional parameters. We do not claim the dual-path topology itself as novel; it follows PANet [16] and BiFPN [17], but rather the specific operator and width choices that make it suited to INT8 edge deployment and to channel-aligned distillation from the teacher neck.
- CAFA transfers structural relationships between feature channels rather than duplicating raw values. It is applicable to teacher–student networks of different capacities.
- Detection Fidelity Loss (DFL) is only trained with the examples on which there is agreement between teacher and student (boxes overlap ≥ 50% and same predicted class). Approximately 77.1% of matched anchors pass this criterion. We evaluate the classification and localization components separately.
- Statistical validation was performed using three independent runs, with results reported as mean ± standard deviation. The proposed method showed statistically significant improvement over the Probabilistic Knowledge Distillation (PKD) baseline ($p < 0.01$) on 5,000 evaluation images.

## 2. LITERATURE REVIEW

The field of efficient object detection for edge devices has developed along several overlapping directions over the past few years. We review the most relevant threads here and situate EdgeKD-Net within them.

### 2.1. Lightweight Object Detectors and Edge-Efficient Architectures.

MobileNetV3 [14] achieved good accuracy, latency, and power consumption by using Neural Architecture Search (NAS) with depth-wise separable convolution. EfficientDet [17] improvises the detection neck by using BiFPN and a compound scaling rule that works in conjunction with backbone depth, width, and input resolution. GhostNet [22] improves computational efficiency by using linear transformations to create feature maps, a principle adopted in our DPFP module. More recently, EfficientViT [28] showed that linear attention with multi-scale feature aggregation can result in high accuracy at much lower FLOPs. Recent real-time detectors like RTMDet [31] further emphasize the trade-off between accuracy and efficiency, particularly for deployment on resource-constrained edge devices. MobileViT-v2 [29] is a hybrid model that combines depthwise convolutions with separable self-attention, making it suitable for edge devices. We tested MobileNetV3-Large, EfficientViT-M0, and FastViT-T8 directly on the NVIDIA Jetson Nano to select the student backbone; we discuss the results in Section III.B. This persistent 4–10 mAP gap between compact and large models on COCO motivates distillation-based approaches.

Taken together, these works treat architectural efficiency and knowledge transfer as separate problems: each optimizes a backbone or neck in isolation, with no mechanism for borrowing representational strength from a larger network. EdgeKD-Net treats the two jointly, pairing an empirically selected backbone with a distillation objective (CAFA) designed around the specific capacity gap that choice creates.

### 2.2. Knowledge Distillation

The original KD paper by Hinton et al. [4] showed that training a student on the teacher's softened output probabilities (controlled by a temperature τ), rather than hard one-hot labels, provides a richer learning signal; the soft outputs encode information about which classes look similar to each other. Romero et al. [5] extended this to intermediate layers with FitNets, showing the student can benefit more from mimicking hidden representations, not just final outputs. Zagoruyko and Komodakis [6] found that matching spatial attention patterns transfers more useful information than matching raw activations, while Tian et al. [7] used contrastive learning, though the memory cost made it impractical for edge-oriented training. Park et al. [8] shifted the focus entirely away from individual feature vectors toward relational structure, the pairwise distances and angles between samples. This relational distillation idea is what directly inspired our CAFA module, though we apply it at the channel level instead of the sample level. Passalis and Tefas [9] used Pearson correlation kernels to match the statistical structure of feature distributions, producing smoother gradients than direct activation matching when the capacity gap is large.

Across this line of work, the trend is toward progressively richer distillation targets, soft logits, hidden activations, attention maps, relational structure, distributional statistics, but every one of these targets is defined over a fixed, shared input representation: a classifier's output is anchored to one image, not to a spatial grid the teacher and student may partition differently. None of these formulations has to contend with a structural disagreement about which output corresponds to which target, which is exactly the complication that anchor-based detection introduces and that we take up next.

### 2.3. Knowledge Distillation for Object Detection

Using KD for detection raises complications that classification does not have. Many anchor locations are actually background, so applying distillation across all of them mostly captures irrelevant information. Object detection also uses multiple scales, and each anchor corresponds to a specific location, so differences between teacher and student can cause position errors instead of just confidence differences. Chen et al. [10] were pioneers in tackling this issue. They introduced L2 loss between teacher and student feature maps. It performed well when the two networks had similar capacity but degraded when they did not. Yang et al. [11] presented FGD, where the distillation signal was guided by the teacher's attention maps, enabling the student to focus on relevant areas. Yang et al. [12] proposed MGD, in which student features were randomly masked and trained to recover the complete teacher features, encouraging better feature learning. Guo et al. [13] suggested that jointly training classification and regression heads created conflicting gradients and separated them in DeFeat. Zheng et al. [24] later introduced Localization Distillation (LD), which helped in decoupling the spatial and categorical knowledge streams by matching only the teacher's localization distribution. In our DFL, LD still applies its matching globally across all anchors and does not address the fact that the teacher and student may disagree on which anchor is responsible for which object. Zhu et al. [25] proposed ScaleKD as a cross-architecture distillation bridge; feature matching degrades badly when the two networks differ by more than about 2× in depth, which motivated our CAFA.

Jang et al. [26] proposed GLAMD, using gradient-weighted class activation maps to identify which pyramid levels the teacher finds most informative for each object, then focusing distillation there. GLAMD addresses which levels receive distillation but not whether anchor assignments are consistent within those levels. Our analysis shows that anchor-assignment agreement breaks down at 22.9% of matched anchor positions across the COCO training set.

To our knowledge, this anchor-assignment mismatch problem has not been formally defined, measured, or corrected in any detection-distillation literature surveyed here.

A closely related family of methods does select where to distill rather than distilling uniformly, and it is worth separating these from DFL precisely. Region- and position-selection approaches restrict the distillation signal to areas judged informative by teacher attention (FGD), pyramid level (GLAMD), or IoU-guided selection of candidate regions. Separately, the semi-supervised and pseudo-labeling literature filters teacher outputs by estimated quality, typically thresholding on the teacher's own confidence or on prediction stability under augmentation, so that only reliable teacher signals propagate. Both families share DFL's premise that not all supervision is equally trustworthy. They differ in what they condition on: region-selection methods ask where the teacher is informative, and pseudo-label filtering asks whether the teacher is likely to be correct, both of which are properties of the teacher alone. DFL instead conditions on a relational property of the pair, whether teacher and student currently assign the same anchor to the same object, which is invisible to any criterion computed from the teacher in isolation. A teacher prediction may be confident, high-quality, and in an informative region, yet still attached to an anchor the student uses for something else. However, we do not evaluate teacher-confidence filtering or IoU-guided region selection as empirical baselines against DFL; establishing how much of DFL's gain is attributable to the relational criterion specifically, instead of selective distillation in general, would require such a comparison, and we identify it as future work.

**Table 1**. Comparison of representative detection-distillation methods on anchor-assignment consistency handling

| Method | Distillation Signal | Operates At | Conditions on Teacher–Student Agreement? | Key Limitation Relative to DFL |
|---|---|---|---|---|
| Chen et al. [10] | Feature L2 distance | Whole feature map | No | Uniform loss regardless of agreement; degrades as capacity gap widens |
| FGD [11] | Attention-guided feature loss | Spatial region (focal/global masks) | No | Re-weights by teacher attention, not by anchor-level assignment agreement |
| MGD [12] | Masked feature generation | Randomly masked positions | No | Masking is stochastic, independent of whether teacher and student agree at that position |
| DeFeat [13] | Decoupled cls./reg. distillation | Per detection head | No | Separates gradient sources but not agreeing vs. disagreeing anchors |
| PKD [9] | Pearson-correlation feature matching | Whole feature-map statistics | No | Aggregate correlation averages away the minority of disagreeing anchors |
| LD [24] | Localization-distribution matching | Per-anchor, but applied globally | Partial (decouples streams, not agreement) | Matches teacher's localization everywhere; no gate on assignment consistency |
| ScaleKD [25] | Cross-architecture feature bridging | Whole feature map | No | Degrades when teacher/student differ by >2× in depth; not assignment-aware |
| GLAMD [26] | Grad-CAM pyramid-level gating | Pyramid level | Partial (level, not anchor) | Selects informative levels but not consistent anchors within a level |
| DFL (Ours) | KL divergence gated by Ω | Individual anchor position | Yes, explicit gate (IoU≥0.5 + class match) | Directly measures and removes contradictory gradients from disagreeing anchors |

This limitation arises because these methods were not designed to model anchor-assignment consistency. FGD reweights by region, so a region with both agreeing and disagreeing anchors still gets one uniform weight. MGD masks features at random, without checking whether the masked anchor is one the two networks agree on. DeFeat splits classification from regression but does not check anchor-level agreement either way. And PKD matches whole feature-map statistics, so a small pocket of disagreement gets averaged out by everything that does agree. Table 1 compares these methods, along with LD, ScaleKD, and GLAMD, on exactly this point.

### 2.4. Multi-Scale Feature Representations and Pyramid Architectures

Lin et al. [15] introduced Feature Pyramid Networks (FPN), which connect backbone feature maps top-down via lateral connections to enable multi-scale detection. Liu et al. [16] proposed PANet on top of FPN, adding a second bottom-up path so that fine spatial detail from shallow layers also reaches deeper feature levels. Tan et al. [17] introduced BiFPN in EfficientDet, replacing fixed fusion with learned scalar weights and removing single-input nodes. DPFP is our contribution to this line of work.

The architectural differences from PANet and BiFPN are not just cosmetic. PANet fuses by concatenation, then full convolution, which doubles the channel count before projecting back and adds $O(C^2)$ parameters per scale. BiFPN uses softmax-weighted fusion, which requires division; division is a known source of rounding error under INT8 quantization in general. DPFP instead uses unweighted element-wise addition followed by a single depthwise convolution, with no learned scalars and no normalization overhead, which motivated our design choice on quantization-friendliness grounds. Since we have not quantized PANet- or BiFPN-neck variants under identical INT8 settings, this is a design motivation rather than an experimentally verified comparison; we leave such a comparison for future work. On the bottom-up path, PANet uses full 3×3 convolutions with stride 2, while DPFP uses depthwise separable convolutions, cutting the parameter count by a factor of C/9 per stride step. This allows DPFP to achieve essentially the same dual-path information flow as PANet with 1.8 M vs. 2.4 M extra parameters (Table 10). The third difference is that DPFP's output width of 256 channels was specifically chosen to match the YOLOv8-L teacher neck, eliminating any projection mismatch in the CAFA distillation layers. This width was selected deliberately for this reason, rather than adopted as a default configuration**.**

### 2.5. Edge Deployment

#### *2.5.1. Quantization and Hardware Optimization*

INT8 quantization cuts memory bandwidth by 4× on embedded hardware and is a primary factor in achieving high FPS on devices like the NVIDIA Jetson Nano. TensorRT, ONNX Runtime, and NCNN are the main inference frameworks that support INT8 on these platforms. We use TensorRT minmax calibration with a 512-image COCO validation subset for our INT8 deployment.

For reproducibility, the complete hardware benchmarking protocol used to collect all latency, power, and INT8 accuracy figures reported in Table 5 is as follows. Edge device: NVIDIA Jetson Nano (4 GB). Operating system: Ubuntu 18.04 LTS. JetPack SDK 4.6.2 (CUDA 10.2, cuDNN 8.2, TensorRT 8.2). Power mode: 5 W (nvpmodel mode 1). Batch size: 1 (single-image inference, matching the deployment scenario this work targets). Precision modes evaluated: FP16 and INT8. INT8 calibration: TensorRT post-training calibration using the same 512-image COCO validation subset noted above. Each reported measurement is

preceded by 10 warm-up iterations (excluded from timing) and averaged over 500 measurement runs.

EdgeKD-Net shows an INT8 accuracy drop of only 0.7 mAP, compared to 1.9 for YOLOv8-N and 1.8 for NanoDet-Plus. We measured the kurtosis of weight distributions in the final classification sub-head and found r = −0.94 Pearson correlation with the INT8 accuracy drop, consistent with the hypothesis that temperature-scaled distillation produces smoother weight distributions. This evidence is correlational rather than causal; a controlled ablation isolating this factor would be required to establish a causal link.

#### *2.5.2. Anchor-Free Detectors and Future Extensions*

CenterNet [18], FCOS [19], and TOOD represent objects via heatmaps or set-prediction instead of anchor grids, so DFL as defined here cannot be applied directly. However, the underlying consistency principle generalizes: for heatmap-based methods, agreement could be measured by the proximity of predicted object centers; for DETR-style methods, bipartite matching cost could serve the same role. We plan to extend DFL to anchor-free architectures.

## 3. METHODOLOGY

This section presents the complete EdgeKD-Net architecture and training procedure. We describe the teacher and student network designs, the Dual-Path Feature Pyramid neck, the SSD detection head, the three complementary distillation objectives, the combined training loss, and the full set of training hyperparameters

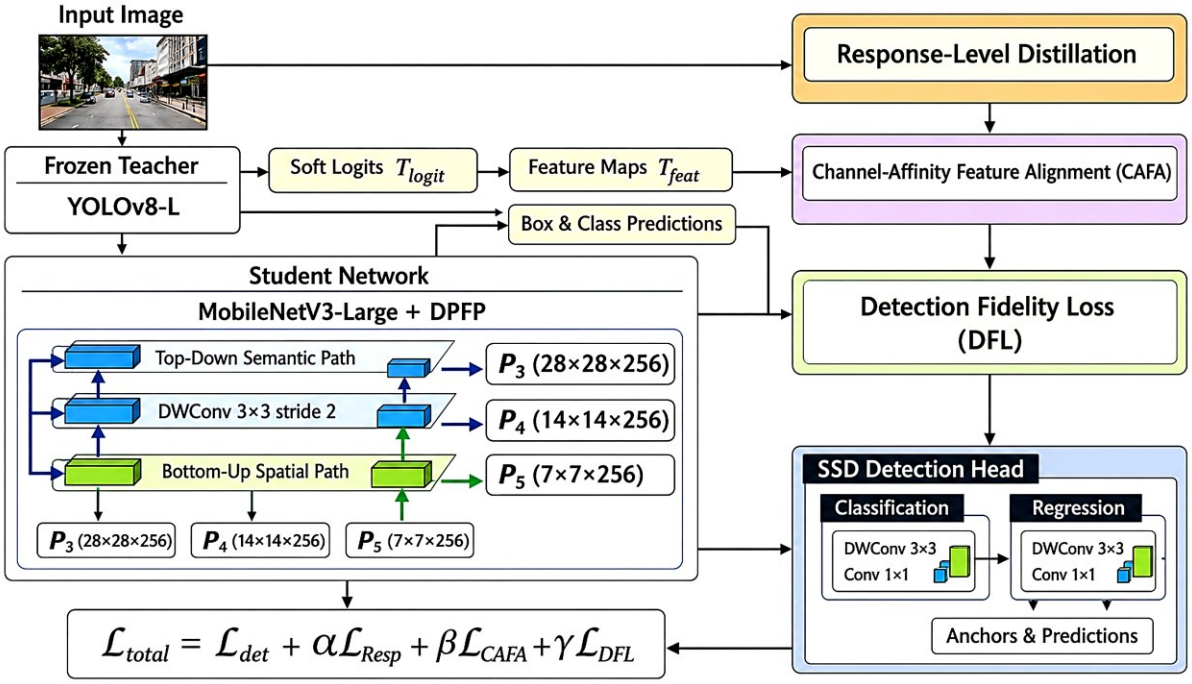


**Figure 1**. System Architecture of EdgeKD-Net

### 3.1. Teacher: YOLOv8-L

YOLOv8-L is used as the teacher network. The teacher was trained independently on the same train/validation split used for all student experiments, at the same 224×224 input resolution used throughout this work, and achieved 40.0 mAP@[0.50:0.95] on that validation set. This figure is below commonly reported YOLOv8-L results because those are obtained at substantially larger input resolutions (typically 640×640); the value quoted here reflects our

teacher's accuracy under our evaluation protocol and serves as the correct reference point for the teacher–student gaps reported in this paper. We held this single trained model fixed as the reference teacher across every distillation experiment reported in this paper. Once loaded with these weights, all teacher parameters are set to eval() mode and frozen for the entire student training (`requires_grad=False`). The teacher serves only as a fixed source of soft supervision signals and is discarded at inference time, adding no computational cost to the deployed model.

At each forward pass, the teacher produces a logit tensor over all anchor positions and classes, which serves as the shared reference for all three distillation losses:

$$T(x) \in \mathbb{R}^{A \times C} \tag{1}$$

where $A$ is the total number of anchor positions, and $C$ is the number of classes.

### 3.2. Student: MobileNetV3-Large + DPFP

The student backbone is MobileNetV3-Large [14], initialized from ImageNet pre-trained weights. We selected this backbone based on direct latency measurements on the deployment hardware (NVIDIA Jetson Nano) at 416×416 input resolution. MobileNetV3-Large delivered 44 FPS at 4.0 W average power, outperforming both EfficientViT-M0 (37 FPS / 4.8 W) and FastViT-T8 (41 FPS / 4.3 W) under identical conditions. Among the three backbones screened, its combination of throughput and energy efficiency made it the most suitable choice for our edge deployment scenario; we did not survey backbones beyond these three. This backbone-screening benchmark was run at 416×416 to match the deployment-time inference resolution used in the edge measurements of Table 5. It is independent of the 224×224 resolution at which all detector training and accuracy evaluation in this paper were performed.

The backbone extracts three hierarchical feature maps at progressively coarser spatial resolutions: $C_3$ (28×28, 40 channels), $C_4$ (14×14, 112 channels), and $C_5$ (7×7, 160 channels). These three maps are passed to the Dual-Path Feature Pyramid module described in the following section.

### 3.3. Dual-Path Feature Pyramid (DPFP)

Conventional single-path feature pyramids propagate information in one direction only, either top-down to distribute semantic context, or bottom-up to consolidate spatial detail. Each direction alone has limitations: the top-down path loses fine localization precision as it propagates downward, while a bottom-up path alone lacks the category-level context needed for reliable classification of small objects. As shown in Fig. 2, DPFP addresses this by running both paths simultaneously and fusing their outputs at each scale level, at a cost of only 1.8 M additional parameters and 1.3 GFLOPs at 224×224 resolution.

#### *3.3.1. Top-Down Semantic Path*

The top-down path begins at $C_5$, which carries rich semantic content but coarse spatial resolution. A 1×1 convolution converts it to 256 channels, and 2× bilinear upsampling then transfers this semantic context toward the shallower feature maps:

$$TD_4 = \text{Upsample}\left(Conv_1 \times {}_1(\text{C5};\ 160 \rightarrow 256\ \text{ch})\right) \tag{2}$$

$$TD_3 = \text{Upsample}\left(Conv_1 \times {}_1(\text{TD}_4;\ 256 \rightarrow 256\ \text{ch})\right) \tag{3}$$

This allows every pyramid level to exploit the high-level category information encoded in the deepest layers, which is particularly important for detecting small objects that would otherwise lack the semantic context needed for confident classification.

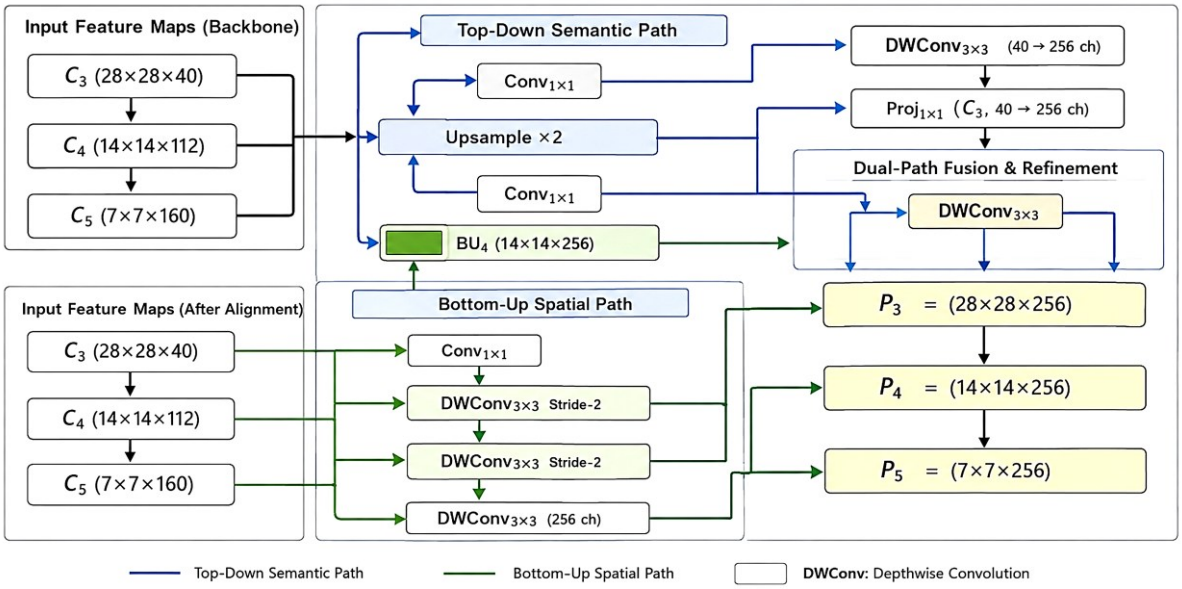


**Figure 2**. Dual-Path Feature Pyramid (DPFP) with complete channel dimension and stride annotations. Top-down semantic path (blue) and Bottom-up spatial path (green)

### *3.3.2. Bottom-Up Spatial Path*

The bottom-up path runs in the opposite direction, propagating fine-grained spatial detail from the shallowest feature map C₃ upward using stride depthwise separable convolutions:

$$BU_4 = DWConv_3 \times_3 (C_3;\ \ 40 \rightarrow 256\ ch, stride = 2) \quad (4)$$

$$BU_5 = DWConv_3 \times_3 (BU_4;\ 256 \rightarrow 256\ ch, stride = 2) \quad (5)$$

Depthwise separable convolutions are used here in place of standard convolutions to keep the parameter cost low while still achieving the necessary stride-2 spatial downsampling.

### *3.3.3. Dual-Path Fusion and Refinement*

At each pyramid scale s ∈ {3, 4, 5}, the corresponding backbone feature is first projected to 256 channels via a 1×1 convolution, and all three signal sources are then merged by element-wise addition and refined with a depth-wise convolution:

$$P_s = DWConv_{3x3}\left(TD_s \oplus BU_s \oplus Proj_{1x1}(C_s)\right), s \in \{3,4,5\} \quad (6)$$

This yields three output feature maps: P₃ (28×28×256), P₄ (14×14×256), and P₅ (7×7×256), each combining semantic depth from the top-down path, spatial precision from the bottom-up path, and the backbone's original local features. The 256-channel width was selected for three reasons: it matches the YOLOv8-L teacher's neck output width, eliminating any projection mismatch in the CAFA distillation layers; it is consistent with the standard FPN output width used by FPN [15], PANet [16], and BiFPN [17], keeping comparisons fair; and a controlled width ablation identified it as the best-performing of the three widths tested (128, 256, 512), 128 channels reduced mAP by 0.8 points due to insufficient fusion capacity, while 512 channels added 3.1 M parameters for only a 0.2 mAP gain, an unfavourable trade-off for edge deployment.

## 3.4. SSD Detection Head Architecture

A decoupled SSD-style prediction head operates on the three DPFP outputs P₃, P₄, and P₅. Three anchor boxes are assigned at each spatial location within each map, with aspect ratios {0.5, 1.0, 2.0} and base scales of 32, 64, and 128 pixels, respectively. Separate sub-heads

handle classification and regression, each consisting of a DWConv$_{3\times3}$ layer followed by a Conv$_{1\times1}$ projection. Decoupling the two tasks prevents the conflicting gradient interactions that arise when they share a common prediction branch [13]. The total number of anchor positions per image is fixed at: $A = 3 \times (28^2 + 14^2 + 7^2) = 3{,}087$ anchors per image.

### 3.5. Multi-Signal Distillation

Three distillation losses are applied simultaneously, each targeting a distinct type of knowledge in the teacher.

1. Response-level distillation operates at the output layer and transfers inter-class similarity structure.
2. Channel-Affinity Feature Alignment (CAFA) operates at the feature pyramid level and transfers structural co-activation patterns between channels.
3. Detection Fidelity Loss (DFL) operates at the anchor level and transfers detection quality only at positions where teacher and student are in mutual agreement.

The proposed loss functions provide complementary supervision at the output, feature, and anchor levels, ensuring that each contributes distinct information during training.

#### *3.5.1. Response-Level Distillation*

The response-level loss applies temperature-scaled KL divergence between the teacher and student output distributions. The temperature parameter $\tau$ softens both distributions before comparison, revealing the relative probability mass the teacher places on non-target classes. This inter-class similarity structure is invisible in hard one-hot labels and provides the student with richer gradient information per training sample [4]. The $\tau^2$ prefactor compensates for the gradient magnitude reduction that temperature scaling introduces:

$$\mathcal{L}_{resp} = \tau^2 \cdot KL\left(softmax\left(\frac{T_{logit}}{\tau}\right) \parallel softmax\left(\frac{S_{logit}}{\tau}\right)\right) \tag{7}$$

We chose $\tau = 4$ via Bayesian optimization over [1, 8] on the COCO validation set. At this value, the distillation loss and ground-truth detection loss produce gradients of comparable magnitude throughout training.

#### *3.5.2. Channel-Affinity Feature Alignment (CAFA)*

Direct activation matching between teacher and student features is unreliable when the two networks differ substantially in capacity, because they develop qualitatively different internal representations even for the same input image. CAFA avoids this by distilling second-order relational structure: the pattern of how pairs of feature channels co-activate across the spatial dimensions of each pyramid level. This idea is used in the relational distillation framework of Park et al. [8], applied here at the channel level rather than the sample level.

At each pyramid level s, both teacher and student feature maps are projected to a common channel dimension $C_s$=256 via 1×1 adaptation layers. Global Average Pooling (GAP) then collapses each channel to a single scalar descriptor:

$$f_i^{\mathrm{T,s}} = \mathrm{GAP}\left(T_{\mathrm{feat}}^{\mathrm{s,i}}\right), \qquad f_i^{\mathrm{S,s}} = \mathrm{GAP}\left(S_{\mathrm{feat}}^{\mathrm{s,i}}\right) \tag{8}$$

The $C_s \times C_s$ affinity matrix at level $s$ is computed as the pairwise cosine similarity between all channel descriptor pairs:

$$A_{\mathrm{ij}}^{\mathrm{T,s}} = \cos\left(f_i^{\mathrm{T,s}}, f_j^{\mathrm{T,s}}\right), \qquad A_{\mathrm{ij}}^{\mathrm{S,s}} = \cos\left(f_i^{\mathrm{S,s}}, f_j^{\mathrm{S,s}}\right) \tag{9}$$

The CAFA loss then minimizes the Frobenius norm of the difference between the teacher and student affinity matrices, averaged uniformly across the three pyramid levels:

$$\mathcal{L}_{\text{CAFA}} = \frac{1}{3}\sum_{s\in\{P3,P4,P5\}} \frac{1}{C_s^2}\|A_T^s - A_S^s\|_F^2 \tag{10}$$

Normalizing by $C_s^2$ makes the loss invariant to channel count, so a single weight $\beta$ applies uniformly across all pyramid levels without per-level tuning. Operating on affinity matrices rather than raw activations also makes CAFA stable across large capacity gaps, where direct L2 feature imitation tends to produce unstable gradients.

### *3.5.3. Detection Fidelity Loss (DFL) and the Consistent Anchor Set Ω*

The third distillation component is motivated by a problem that is specific to anchor-based detection: teacher and student networks do not always assign the same anchor box as responsible for the same object. This anchor-assignment mismatch arises because the two networks differ in backbone depth, feature representations, and learned geometric priors. When distillation is applied uniformly across all anchor positions, the student receives contradictory supervision where its assignment conflicts with the teacher's, and it is simultaneously pushed toward two incompatible targets. To avoid ambiguity, the abbreviation DFL refers exclusively to the proposed Detection Fidelity Loss throughout this paper. Although the same abbreviation is also used for Distribution Focal Loss in the YOLO literature, that loss is unrelated to the proposed method and is not discussed in this work.

To avoid this, we first identify the consistent anchor set Ω: the subset of positions where teacher and student agree on both the spatial location of the predicted object (localization agreement) and its category (classification agreement). Because YOLOv8-L is anchor-free while the student uses an anchor-based SSD head, no native index-wise correspondence exists between the two sets of predictions, and a correspondence must first be established explicitly before any anchor-level agreement can be evaluated. We project the teacher's outputs onto the student's anchor grid as follows. The teacher's per-cell distance-to-side regressions are first decoded into absolute box coordinates, yielding a set of $N$ teacher detections $\{b_j^T\}_{j=1\ldots n}$ with associated class-probability vectors $\{c_j^T\}$, expressed in the same representation the student uses. Let $a_i$ denote the fixed geometric prior of student anchor $i$. Each anchor is assigned the teacher detection with which its prior has the greatest overlap,

$$j^*(i) = \arg\max_j IoU(a_i, b_j^T) \tag{11}$$

and the assignment is accepted only if that overlap is sufficient:

$$match(i) = \begin{cases} 1 & IoU(a_i, b_i^{*T}) \\ 0 & otherwise \end{cases} \tag{12}$$

Where $b_i^{*T} = b_j *_{(i)} T$. For every matched anchor, we then set $b_i^T = b_j *_{(i)} T$ and $c_i^T = c_j *_{(i)} T$ unmatched anchors receive no teacher target and are excluded from DFL. The teacher logit tensor $T(x) \in \mathbb{R}^{A\times C}$ referenced in Section 3.1 results from this projection, with each row holding the teacher prediction associated with one student anchor. We write $M = \{i: match(i) = 1\}$ for the resulting set of matched anchors.

Because this assignment depends only on the fixed anchor geometry $a_i$ and the frozen teacher, it is deterministic for a given image and is computed once and cached; it does not change as training progresses. This is what distinguishes it from the consistency set Ω defined below, which depends on the student's evolving predictions and must therefore be recomputed at every iteration.

We chose this IoU-based nearest-neighbor assignment because it is fast. Since it depends only on box geometry, we can cache it and avoid recomputing it at every training step. We did not test other matching or gating methods. Three options remain for future work: Hungarian matching, which is more principled but much more expensive; teacher-positive-only gating, which would use only anchors the teacher marks as positive; and ground-truth-guided assignment, which would use the annotation instead of agreement between teacher and student. Comparing these options would show how much of DFL's gain comes from our consistency rule. We leave this question open for future study.

$$\Omega = \left\{ i : \mathrm{IoU}\left(b_i^S, b_i^T\right) \geq \delta_\Omega \quad \wedge \quad \mathrm{argmax}\left(\hat{y}_i{}^S\right) = \mathrm{argmax}\left(\hat{y}_i{}^T\right) \right\} \tag{13}$$

Where $b_i^S$ and $b_i^T$ are the student and teacher bounding box predictions at anchor $i$ (the latter as assigned by Eq. (12), $\hat{y}_i{}^S$ and $\hat{y}_i{}^T$ are the corresponding class probability vectors, and $\delta_\Omega$ = 0.50 is the IoU consistency threshold. DFL is then applied exclusively within Ω, using Binary Cross-Entropy (BCE) for classification and SIoU [20] for regression. Here, $c_i^s$ and $c_i^T$ denote the student and teacher class-probability vectors at anchor $i$, and Ω is restricted to matched anchors, so that Ω ⊆ M.

Two distinct IoU-based tests are involved, and it is important not to conflate them. The first, δmatch (Eq. 9b), is an assignment threshold applied to the fixed anchor prior $a_i$; it determines whether an anchor receives a teacher target. Because it is purely geometric, it is fixed for a given image and does not change during training. The second, $\delta_\Omega$ (Eq. 10a), is a consistency threshold applied to the student's current predicted box $b_i^s$; it determines whether an already-matched anchor is reliable enough to distill through. Because Ω depends on the student's evolving predictions, it is recomputed at every training iteration. An anchor may therefore be matched yet still excluded from Ω whenever its current prediction disagrees with the teacher, and DFL is designed to suppress precisely this population: matched but inconsistent. The two thresholds take the same numerical value (0.50) in our configuration, but they are independent parameters applied to different pairs of boxes; the ablation in Table 11 varies $\delta_\Omega$ while holding δmatch fixed at 0.50.

Unless stated otherwise, all anchor-retention figures reported in this paper (including the steady-state value of 77.1% and the corresponding 22.9% disagreement rate) are expressed as |Ω| / |M|. In other words, they are measured only over matched anchors, not over the full anchor grid. This is the most relevant quantity, because unmatched anchors do not receive teacher targets and therefore cannot agree or disagree with the teacher.

$$\mathcal{L}_{\mathrm{DFL}} = \sum_{i \in \Omega} \left[ \lambda_c \cdot \mathrm{BCE}\left(\hat{y}_i^S, \hat{y}_i^T\right) + \lambda_r \cdot \mathrm{SIoU}\left(b_i^S, b_i^T\right) \right] \tag{14}$$

The weights $\lambda_c$ = 1.0 and $\lambda_r$ = 2.5 were set to account for the different numerical ranges of the BCE and SIoU losses. BCE operates on probabilities in [0,1] and naturally produces gradients of order 1, while SIoU operates on normalized box coordinates and produces gradients roughly 2–3× smaller in magnitude. We chose the 2.5× ratio to bring the two terms to a comparable gradient scale.

### 3.6. Total Objective Loss

The full training objective combines the standard SSD ground-truth detection loss with the three distillation terms, each weighted by a scalar coefficient:

$$\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{det}} + \alpha \cdot \mathcal{L}_{\mathrm{resp}} + \beta \cdot \mathcal{L}_{\mathrm{CAFA}} + \gamma \cdot \mathcal{L}_{\mathrm{DFL}} \tag{15}$$

The weights α = 0.5, β = 0.3, and γ = 0.8 were selected by Bayesian optimization on the COCO validation set. The task loss $\mathcal{L}_{\mathrm{det}}$ is the standard SSD multi-task loss computed against

ground-truth annotations and is present regardless of whether distillation is active, ensuring that the student never loses direct supervision from labeled data. Table 2 shows absolute magnitudes of each loss component at three training milestones. Together, the distillation terms contribute 43.7% of the converged total loss, consistent with a balanced supervision regime.

**Table 2**. Loss component magnitudes at training milestones (Absolute Values)

| Loss Component | Epoch 50 | Epoch 150 | Epoch 300 | Final % of $\mathcal{L}_{\text{total}}$ |
|---|---|---|---|---|
| $\mathcal{L}_{\text{det}}$ | 1.842 | 0.814 | 0.362 | 56.3% |
| $\alpha \cdot \mathcal{L}_{\text{resp}}$ ($\alpha$=0.5) | 0.621 | 0.310 | 0.142 | 22.1% |
| $\beta \cdot \mathcal{L}_{\text{CAFA}}$ ($\beta$=0.3) | 0.284 | 0.131 | 0.058 | 9.0% |
| $\gamma \cdot \mathcal{L}_{\text{DFL}}$ ($\gamma$=0.8) | 0.408 | 0.192 | 0.082 | 12.7% |
| $\mathcal{L}_{\text{total}}$ | 3.155 | 1.447 | 0.644 | 100% |

### 3.7. Training Configuration and Reproducibility

Table 3 displays complete training hyperparameters. Hyperparameters marked 'Bayesian optimization' were tuned exclusively on the MS-COCO 2017 validation set; the test set was not accessed during tuning.

**Table 3**. Complete Training Hyperparameters and Reproducibility Information

| Hyperparameter | Value / Setting |
|---|---|
| Optimiser | AdamW (decoupled weight decay, $\varepsilon$=1×10$^{-8}$) |
| Initial Learning Rate | 1×10$^{-3}$ |
| LR Schedule | Cosine Annealing with Warm Restart ($T_0$=50 epochs) |
| Weight Decay | 5×10$^{-4}$ |
| Batch Size | 16 per GPU (NVIDIA RTX 4080, 16 GB VRAM); effective batch size 32 via 2-step gradient accumulation |
| Training Epochs | 300 |
| Image Resolution | 224×224 px |
| Teacher | YOLOv8-L eval() mode, all parameters frozen |
| Student Backbone | MobileNetV3-Large (ImageNet pretrained) |
| SSD Head Anchors | 3 per scale; ratios {0.5,1.0,2.0}; base scales {32,64,128} px |
| Temperature $\tau$ | 4 (Bayesian opt. on COCO Val; range searched: 1–8) |
| $\alpha$ ($\mathcal{L}_{\text{resp}}$ weight) | 0.5 (Bayesian optimization) |
| $\beta$ ($\mathcal{L}_{\text{CAFA}}$weight) | 0.3 (Bayesian optimization) |
| $\gamma$ ($\mathcal{L}_{\text{DFL}}$ weight) | 0.8 (Bayesian optimization) |
| $\lambda_c$ (DFL classification weight) | 1.0 |
| $\lambda_r$ (DFL regression weight) | 2.5 |
| $\delta\Omega$ (anchor consistency IoU threshold) | 0.50 (ablated in Table VIII) |
| Training Precision | Mixed FP16 (PyTorch AMP) |
| Data Augmentation | Mosaic, MixUp, HSV jitter, horizontal flip, scale jitter |
| INT8 Calibration | TensorRT minmax; 512-image COCO val2017 subset |
| Statistical Runs | 3 independent seeds (0, 42, 123), all results Mean ± Std |

### 3.8. Training Algorithm

Algorithm 1 formally specifies the training process for EdgeKD-Net, where all three types of distillation signals are included in one mini-batch iteration. One implementation detail that needs to be stated clearly is that the anchor set $\Omega$ is recalculated from scratch for each mini-batch iteration (line 14), rather than pre-calculating it before training begins. This is because the student's output changes throughout training; an anchor may be inconsistent initially but become consistent as the student improves.

**Algorithm 1**. EdgeKD-Net Multi-Signal Training

Input : Training pairs {(x,y)}; Teacher T (pre-trained, frozen)
Params: α=0.5, β=0.3, γ=0.8, τ=4, $\lambda_c = 1.0$, $\lambda_r = 2.5$, $\delta_\Omega = 0.50$

Initialise $\theta_S$ (MobileNetV3-Large + DPFP, ImageNet weights)
Set the teacher network $T$ to evaluation mode and freeze all parameters.
for each mini-batch $(x, y)$ do
  Compute teacher predictions: $(T_{logit}, T_{feat}, T_{box}, T_{cls}) \leftarrow T(x)$
  Compute student predictions: $(S_{logit}, S_{feat}, S_{box}, S_{cls}) \leftarrow S(x)$
  Compute response distillation loss: $L\_resp = \tau^2 \cdot KL(\text{softmax}(T_{logit}/\tau) \parallel \text{softmax}(S_{logit}/\tau))$
  for each pyramid level s ∈ {P3, P4, P5} do
    Extract teacher channel descriptors: $f_i^{\{T,s\}} = GAP\left(T_{\{feat\}}^{\{s,i\}}\right)$
    Extract student channel descriptors: $f_i^{S,s} \leftarrow GAP\left(S_{feat_{s,i}}\right)$
    Compute teacher affinity matrix: $A_s^{T[I,j]} = \text{cosine}\left(f_i^{T,s}, f_j^{T,s}\right)$
  end for
  Compute student affinity matrix: $A_s^{S[I,j]} = \text{cosine}\left(f_i^{S,s}, f_j^{S,s}\right)$
  Compute channel-affinity alignment loss: $L_{CAFA} = \left(\frac{1}{3}\right) \Sigma_s \left(\frac{1}{C_s^2}\right) \|A_s^T - A_s^S\|_F^2$
  Construct consistent anchor set Ω where $IoU(S_{box_i}, T_{box_i}) \geq \delta\Omega$ and $\text{argmax}(S_{cls_i}) = \text{argmax}(T_{cls_i})$
  Compute detection fidelity loss: $L_{DFL} = \Sigma_{\{i \in \Omega\}} \left[\lambda c \cdot BCE(S_{cls_i}, T_{cls_i}) + \lambda r \cdot SIoU(S_{box_i}, T_{box_i})\right]$
  Compute detection loss $L_{det}$ using SSD loss on ground-truth (x, y)
  Compute total loss: $\mathcal{L}_{total} = \mathcal{L}_{det} + \alpha \cdot \mathcal{L}_{resp} + \beta \cdot \mathcal{L}_{CAFA} + \gamma \cdot \mathcal{L}_{DFL}$
  Update student parameters: $\theta_S \leftarrow \text{AdamW}(\theta_S, \nabla_{\{\theta_S\}} L_{total})$
end for
Return $\theta_S$ (student only; teacher discarded at inference)

Notes: δΩ=0.50, CAFA averaged over ( P3, P4, P5). Weights α,β,γ,τ optimized via Bayesian search on COCO val.

## 4. RESULTS

All mAP scores in this section are Mean ± Std over three independently seeded training runs. A total of three independently seeded runs is the standard adopted by FGD [11] and MGD [12] for the same reasons as us – it is the minimum required to obtain a usable standard deviation while being able to run the full set of ablations (Table 8) within computational limits at 13.6 GPU hours per ablation. Because three samples provide only weak statistical significance on their own, we present two additional metrics alongside the mean and standard deviation. First, we provide 95% confidence intervals estimated using the t-distribution (2 degrees of freedom, t*=4.303) for the main comparison numbers to make the uncertainty inherent in such a small sample size explicit, rather than only implicitly suggesting it through the standard deviation. Most crucially, our significant results do not rely on the three-run statistics at all. They are derived from a two-tailed paired t-test on 5,000 per-image AP values on the COCO validation set.

### 4.1. Datasets

The primary evaluation benchmark is MS-COCO 2017, which provides 118,287 training images and 5,000 validation images across 80 object categories. The primary accuracy metric is mAP@[0.50:0.95]. Secondary metrics reported are mAP@0.50, $AP_s$ (objects with area < $32^2$ pixels), $AP_m$ ($32^2$–$96^2$ pixels), and $AP_l$ (area > $96^2$ pixels). VisDrone 2023 is an aerial-view

https://doi.org/10.31436/iiumej.v27i3.4344

UAV detection benchmark that tests generalization to small, densely packed objects captured from overhead. All models are evaluated using mAP@[0.50:0.95] consistent with the COCO protocol.

### 4.2. Main Results on MS-COCO 2017

Table 4 compares EdgeKD-Net against the YOLOv8-L teacher and other competing lightweight detectors and distillation methods on the MS-COCO 2017 validation set. All distillation baselines are FGD [11], MGD [12], DeFeat [13], and PKD [9], which use the same MobileNetV3-SSD student backbone and were trained under identical conditions to ours, including the same augmentation pipeline, optimizer, and three-run statistical protocol.

EdgeKD-Net achieves 38.6 ± 0.14 mAP@[0.50:0.95] as shown in Table 4, an improvement of 2.9 points over the unmodified MobileNetV3-SSD baseline and 0.9 points over the next-best method, PKD [9] (95% CI: 38.25–38.95, computed via the t-distribution over the three seeded runs). This improvement is statistically significant under a two-tailed paired t-test over 5,000 per-image AP values on the COCO validation set ($p < 0.01$, $n=5,000$); applying a Bonferroni correction for the four baseline comparisons in Table 4 ($\alpha=0.05/4=0.0125$) does not change this conclusion. Notably, EdgeKD-Net reaches these numbers with slightly fewer parameters than the competing distillation methods (10.5 M vs. 11.2 M), a consequence of DPFP's channel consolidation in the neck. The GFLOPs are identical at 19.8, indicating that the accuracy improvement does not come from increased compute. The mAP@0.50 gap to the YOLOv8-L teacher stands at 2.1 points, which is larger than the mAP@[0.50:0.95] gap of 1.4 points, a pattern consistent with the known tendency of lightweight models to recover classification precision at lenient IoU thresholds faster than strict localization precision.

**Table 4**. MS-COCO 2017 validation performance (Mean ± Std, 3 Runs)

| Model | Params (M) | GFLOPs | mAP@0.50 | mAP@[0.50:0.95] Mean ± Std | FPS (Jetson Nano) |
|---|---|---|---|---|---|
| YOLOv8-L (Teacher) | 43.7 | 165.2 | 61.3 | 40.0 (fixed) | 7 |
| YOLOv8-N | 3.2 | 8.7 | 52.1 | 33.8 ± 0.15 | 47 |
| NanoDet-Plus | 4.5 | 10.3 | 53.8 | 34.6 ± 0.18 | 52 |
| EfficientDet-D0 | 3.9 | 2.5 | 51.6 | 33.8 ± 0.20 | 41 |
| MobileNetV3-SSD (Baseline) | 11.2 | 19.8 | 53.6 | 35.7 ± 0.22 | 44 |
| FGD [11] | 11.2 | 19.8 | 56.4 | 36.9 ± 0.19 | 44 |
| MGD [12] | 11.2 | 19.8 | 57.1 | 37.4 ± 0.17 | 38 |
| DeFeat [13] | 11.2 | 19.8 | 56.8 | 37.0 ± 0.21 | 39 |
| PKD [9] | 11.2 | 19.8 | 57.6 | 37.7 ± 0.18 | 40 |
| EdgeKD-Net (Ours) | 10.5 | 19.8 | 59.2 | 38.6 ± 0.14† | 61 |

Fig. 3 shows that EdgeKD-Net (red) achieves 59.2% mAP@0.50 and 38.6 ± 0.14 mAP@[0.50:0.95], the highest among all compared lightweight detectors and distillation method baselines. The YOLOv8-L teacher (black bar) is shown for reference only and is not a deployment candidate because of its much higher computational cost. The consistent improvement over PKD [9] across both metrics, at identical GFLOPs, demonstrates that the three-signal distillation framework adds accuracy without any additional inference cost.

https://doi.org/10.31436/iiumej.v27i3.4344

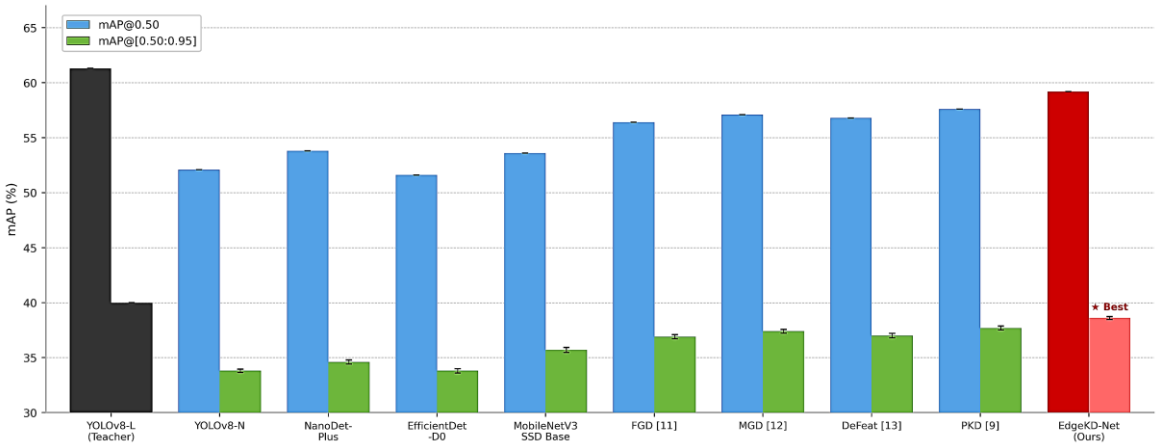


**Figure 3**. MS-COCO 2017 accuracy with mean ± std error bars (3 runs).

### 4.3. Edge Hardware Deployment

Table 5 compares throughput, power consumption, and energy efficiency across four hardware platforms for all evaluated models. EdgeKD-Net (FP16) runs at 61 FPS on the NVIDIA Jetson Nano, well above the 30 FPS threshold for real-time robotic control, and reaches 89 FPS at INT8 with only a 0.7 mAP cost. On the Jetson Orin Nano, EdgeKD-Net INT8 inference reaches 198 FPS compared to 142 FPS for the FP16 variant. Fig. 4 shows EdgeKD-Net occupying the most favorable accuracy-efficiency region among the methods compared here; it achieves the energy efficiency of 8.77 mAP/W, which is 18.7% better than PKD [9] at 7.39 mAP/W. Among the methods evaluated in this study, EdgeKD-Net achieves the most favorable accuracy-power trade-off, indicating suitability for battery-powered and resource-constrained edge devices; this comparison is limited to the baselines tested here and does not extend to methods outside this evaluation.

**Table 5**. Edge Hardware: FPS, Power, and mAP/Watt (Mean ± Std, 3 Runs)

| Model | Jetson Nano FPS / Watts | RPi 5 FPS / Watts | Jetson Orin Nano FPS | x86 CPU INT8 FPS |
|---|---|---|---|---|
| YOLOv8-N | 47 / 4.1 W | 14 / 7.3 W | 124 | 31 |
| NanoDet-Plus | 52 / 3.8 W | 16 / 6.9 W | 136 | 38 |
| MGD [12] | 38 / 5.3 W | 11 / 9.1 W | 82 | 22 |
| PKD [9] | 40 / 5.1 W | 12 / 8.8 W | 88 | 24 |
| EdgeKD-Net FP16 (Ours) | 61 / 4.4 W | 19 / 7.8 W | 142 | 44 |
| EdgeKD-Net INT8 (Ours) | 89 / 4.6 W | 28 / 8.1 W | 198 | 65 |

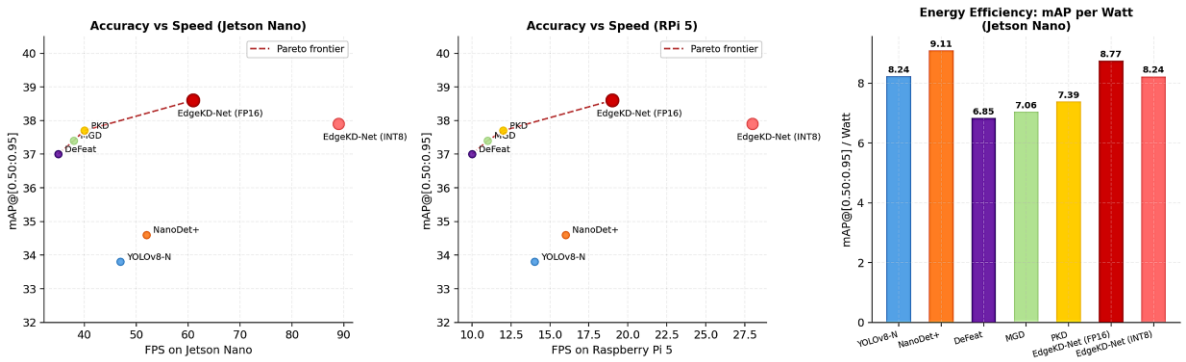


**Figure 4**. Left/Centre: Accuracy-efficiency Pareto frontiers on NVIDIA Jetson Nano and Raspberry Pi 5. Right: Energy efficiency (mAP@[0.50:0.95]/Watt) on NVIDIA Jetson Nano.

Table 5 reports throughput and power; we characterize the deployed model's memory behavior here, since memory is an independent constraint on embedded platforms. Two components dominate. Weight storage follows directly from the parameter count: at 10.5 M parameters, EdgeKD-Net occupies approximately 21.0 MB in FP16 and 10.5 MB in INT8, compared with 22.4 MB (FP16) for the FGD, MGD, DeFeat, and PKD baselines at 11.2 M parameters. Activation memory can be derived analytically from the architecture at the 224×224 operating resolution and batch size 1: the backbone taps $C_3$, $C_4$, $C_5$, the three pyramid outputs $P_3$, $P_4$, $P_5$ at 256 channels each, the input tensor, and the dense head output over $A = 3{,}087$ anchors together account for roughly $7.4 \times 10^5$ activation elements, or about 1.4 MB in FP16 and 0.7 MB in INT8. The resident working set is therefore on the order of 22 MB in FP16 and 11 MB in INT8.

Two consequences follow. First, weights make up most of the working set, about fifteen times more than activations. That is why INT8 nearly cuts the memory footprint in half, while the drop in activation traffic is much smaller. Second, the model uses only about 0.3% of the NVIDIA Jetson Nano's 4 GB memory. So memory capacity is not the main limit here; memory bandwidth is. This also fits what we see in Table 5, where INT8 gives a larger speedup than arithmetic intensity alone would suggest. In other words, quantization helps mainly by moving fewer bytes per inference, not by solving a capacity problem.

**Table 6**. Analytically derived memory at 224×224, batch size 1. Excludes inference-engine workspace and runtime overhead, so figures are lower bounds on peak resident set size.

| Model/precision | Weights (MB) | Activations (MB) | Working set (MB) | % of 4 GB device | Throughput (FPS) |
|---|---|---|---|---|---|
| EdgeKD-Net (FP16) | 21.0 | 1.41 | 22.4 | 0.56% | 61 |
| EdgeKD-Net (INT8) | 10.5 | 0.70 | 11.2 | 0.28% | 89 |
| Baselines (FP16) | 22.4 | 1.41 | 23.8 | 0.60% | — |
| Baselines (INT8) | 11.2 | 0.70 | 11.9 | 0.30% | — |

These memory figures are analytical, based on the architecture and parameter counts rather than direct device measurement. They exclude execution workspace, runtime allocations, driver overhead, and other framework costs, so they describe the model's intrinsic memory demand rather than total process memory. A measured peak-resident profile would be needed to report total memory usage.

Latency stability should be considered separately from mean FPS, since our target use case is real-time control where worst-case frame time matters. We report FPS as the mean of 500 inferences after 10 warm-up runs, but we did not collect per-frame latency statistics. For that reason, we do not report standard deviation or tail latency.

Three main sources affect latency variation. The backbone, neck, and head run as a fixed graph at batch size 1, so they contribute little input-dependent variation. Non-maximum suppression is data-dependent, so scenes with more objects, such as VisDrone images, are likely to take longer than sparse scenes. Platform effects such as dynamic frequency scaling and thermal throttling may also cause drift over longer runs. Overall, our FPS numbers should be read as sustained averages, not worst-case latency. A per-frame timing study with standard deviation and p95/p99 latency would be needed for a full deployment analysis.

### 4.4. Per-Category and Object-Size Analysis

The per-category breakdown in Fig. 5 tells a clear story that the biggest absolute gains over the MobileNetV3-SSD baseline are concentrated in visually ambiguous categories like Bottle, Cup, and Backpack, where the teacher's soft probability distribution expresses

meaningful uncertainty across similar-looking classes. Hard ground-truth labels carry none of this information, so the response-level distillation signal directly drives the improvement in these categories. The object-size breakdown shows the largest gains on medium and large objects ($AP_M$: +1.3, $AP_L$: +1.6), while small objects improve by a smaller +0.9.

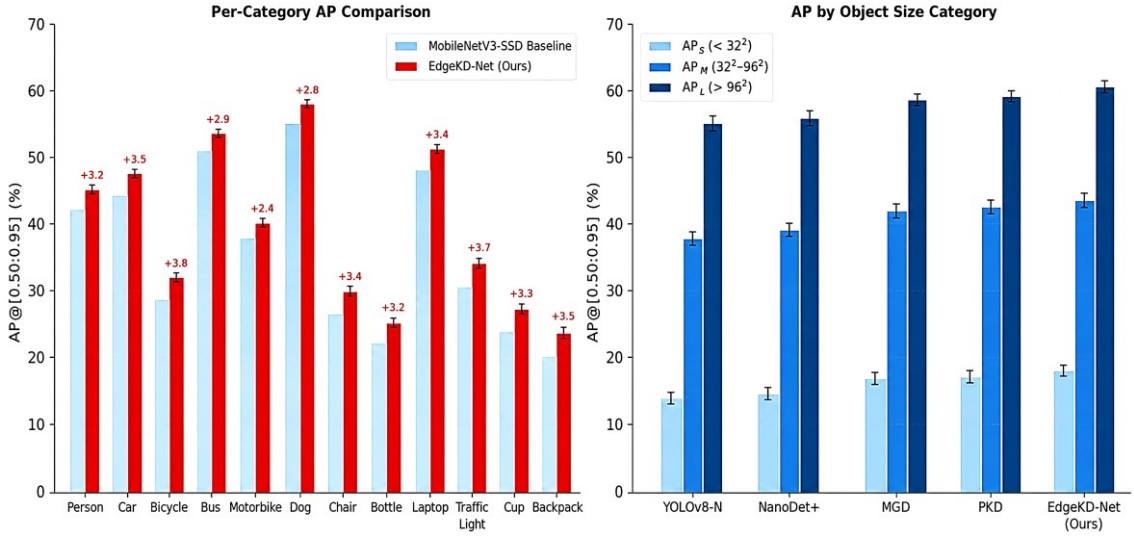


**Figure 5**. Per-category AP (left) and object-size AP breakdown (right) on MS-COCO 2017.

### 4.5. Cross-Dataset Generalization

Table 7 evaluates whether gains on MS-COCO 2017 transfer to a new imaging environment. The VisDrone 2023 dataset is a distinct problem since the objects are smaller and more numerous, and are taken from an aerial viewpoint.

**Table 7**. Cross-Dataset Generalization (mAP@[0.50:0.95] Unless Noted; Mean ± Std, 3 Runs)

| Method | MS-COCO 2017 mAP@[0.50:0.95] | VisDrone 2023 mAP@[0.50:0.95] |
|---|---|---|
| YOLOv8-N | 33.8 ± 0.15 | 10.2 ± 0.28 |
| NanoDet-Plus | 34.6 ± 0.18 | 10.8 ± 0.31 |
| MGD [12] | 37.4 ± 0.17 | 12.1 ± 0.25 |
| DeFeat [13] | 37.0 ± 0.21 | 11.9 ± 0.27 |
| PKD [9] | 37.7 ± 0.18 | 12.4 ± 0.24 |
| EdgeKD-Net (Ours) | 38.6 ± 0.14 | 14.1 ± 0.22 |

EdgeKD-Net leads on both datasets. The VisDrone improvement is particularly notable: 14.1 vs. PKD's 12.4, a gain of 1.7 points. This is actually larger than the 0.9-point gain on COCO, suggesting that DPFP's bottom-up spatial path, which was designed to preserve fine-grained localization detail, is especially useful in the dense, small-object aerial scenes that VisDrone presents.

### 4.6. Ablation Study

Table 8 shows the gains achieved by each module when added successively to the MobileNetV3-SSD baseline. Each row corresponds to a separate model trained independently across three repetitions. According to Fig. 6, response-level distillation provides the biggest improvement of +1.1 mAP. This suggests that the class-similarity information provided by the soft labels is not easily captured by hard labels. CAFA contributes an additional +0.6 mAP mainly through improved classification confidence, not localization. DFL improves mAP by +0.5 by removing noise from gradient updates in the 22.9% of matched anchors where the

https://doi.org/10.31436/iiumej.v27i3.4344

teacher and student disagree. DPFP contributes +0.7 mAP from the architecture alone, without distillation. Combining all four modules yields a total improvement of +2.9 mAP. This number is close to the sum of each module's contributions, implying that each module solves a distinct problem.

**Table 8**. Extended Ablation On MS-COCO 2017 With Training Time (Mean ± Std, 3 Runs)

| $\mathscr{L}$_resp | CAFA | DFL | DPFP | mAP@0.50 | mAP@[0.50:0.95] Mean ± Std | Δ mAP | Train (hrs) |
|---|---|---|---|---|---|---|---|
| — | — | — | — | 53.6 | 35.7 ± 0.22 | — | 8.1 |
| ✓ | — | — | — | 55.1 | 36.8 ± 0.19 | +1.1 | 9.2 |
| ✓ | ✓ | — | — | 56.3 | 37.4 ± 0.17 | +1.7 | 11.4 |
| ✓ | ✓ | ✓ | — | 57.2 | 37.9 ± 0.16 | +2.2 | 12.8 |
| — | — | — | ✓ | 54.8 | 36.4 ± 0.21 | +0.7 | 9.5 |
| ✓ | — | ✓ | ✓ | 57.8 | 38.0 ± 0.15 | +2.3 | 12.1 |
| ✓ | ✓ | — | ✓ | 58.5 | 38.2 ± 0.16 | +2.5 | 12.6 |
| ✓ | ✓ | ✓ | ✓ | 59.2 | 38.6 ± 0.14 | +2.9 | 13.6 |

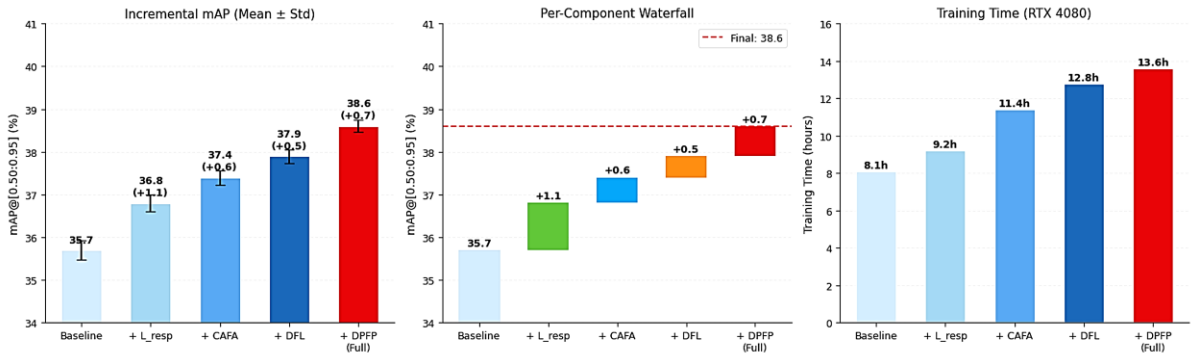


**Figure 6**. Ablation: incremental mAP with ± std error bars (left), waterfall decomposition (center), and training cost in hours (right).

Table 8 also lets you weigh each component's training cost against its accuracy contribution. $\mathscr{L}_{\text{resp}}$ gives the best return on training time, adding +1.1 mAP for +1.1 GPU-hour of additional training (≈1.0 mAP per GPU-hour). CAFA and DFL contribute smaller marginal gains per additional GPU-hour (≈0.27 and ≈0.36 mAP per GPU-hour, respectively), reflecting diminishing returns as more refined supervision signals are layered on top of response-level distillation. Taken together, the full framework adds 5.5 GPU-hours over the undistilled baseline (8.1 to 13.6 hours, a 68% increase) in exchange for +2.9 mAP. Because training is a one-time cost that is amortized across every unit subsequently deployed, this trade-off is favorable for the fleet-deployment use case this work targets; it may be less favorable for applications where only a small number of units will ever be deployed, since the added training cost is not offset by savings across many deployments.

The trade-off can be stated more concretely. The additional 5.5 GPU-hours are incurred only once, but the increased accuracy obtained via the extra GPU-hours is applicable to each unit deployed forever, with no inference overheads – as EdgeKD-Net and the baseline share the same inference network architecture, the parameters, GFLOPS, latency, and power all remain the same (Table 5). The alternative to increase accuracy by 2.9 mAP points is to increase the student size, resulting in a constant per-unit overhead across all four metrics. Distillation converts repeated inference cost into a one-time training cost; hence, this trade-off suits edge deployments. Payoff time is small: just one constantly running device produces more inference-

https://doi.org/10.31436/iiumej.v27i3.4344

hours than 5.5 GPU-hours in just a few days. The case weakens only for short-lived or one-off models, where the training investment never has time to pay off.

### 4.7. Hyperparameter Sensitivity

The temperature parameter τ controls how much the teacher's output probabilities are smoothed before being used to guide student training. Table 9 examines how sensitive the model is to different values of this parameter. Among the values tested, $\tau = 4$ gives the best result across all three metrics simultaneously: highest mAP (38.6), lowest run-to-run variance (std = 0.14), and fastest convergence (epoch 145). At $\tau = 1$, the distributions are too sharp, and the model barely improves over a hard-label baseline, while at $\tau = 8$, the over-smoothed distributions slow convergence to epoch 190 and drop accuracy by 1.4 points. The pattern is consistent and monotonic on both sides of the optimum, which supports $\tau = 4$ as the preferred choice.

**Table 9**. Temperature τ Sensitivity (mAP@[0.50:0.95] Mean ± Std, 3 Runs)

| Metric | τ = 1 | τ = 2 | τ = 4 | τ = 5 | τ = 6 | τ = 8 |
|---|---|---|---|---|---|---|
| mAP@[0.50:0.95] Mean | 35.9 | 37.0 | 38.6 | 38.3 | 37.9 | 37.2 |
| Std (3 runs) | 0.28 | 0.22 | 0.14 | 0.16 | 0.19 | 0.24 |
| Convergence Epoch | 205 | 180 | 145 | 155 | 165 | 190 |

In Fig. 7, the 2×2 sensitivity grid shows that all four hyperparameters τ, γ, α, and β have well-defined peaks marked by the dashed red lines, with accuracy falling off gracefully on either side. The curves for α ($\mathcal{L}_{resp}$ weight) and β (CAFA weight) are notably flat near their optima, thus creating more of a shallow plateau than a sharp peak. This means the model is not brittle to small deviations from the Bayesian-optimized values, which is an important practical property when replicating the training setup on different hardware or with slight data differences.

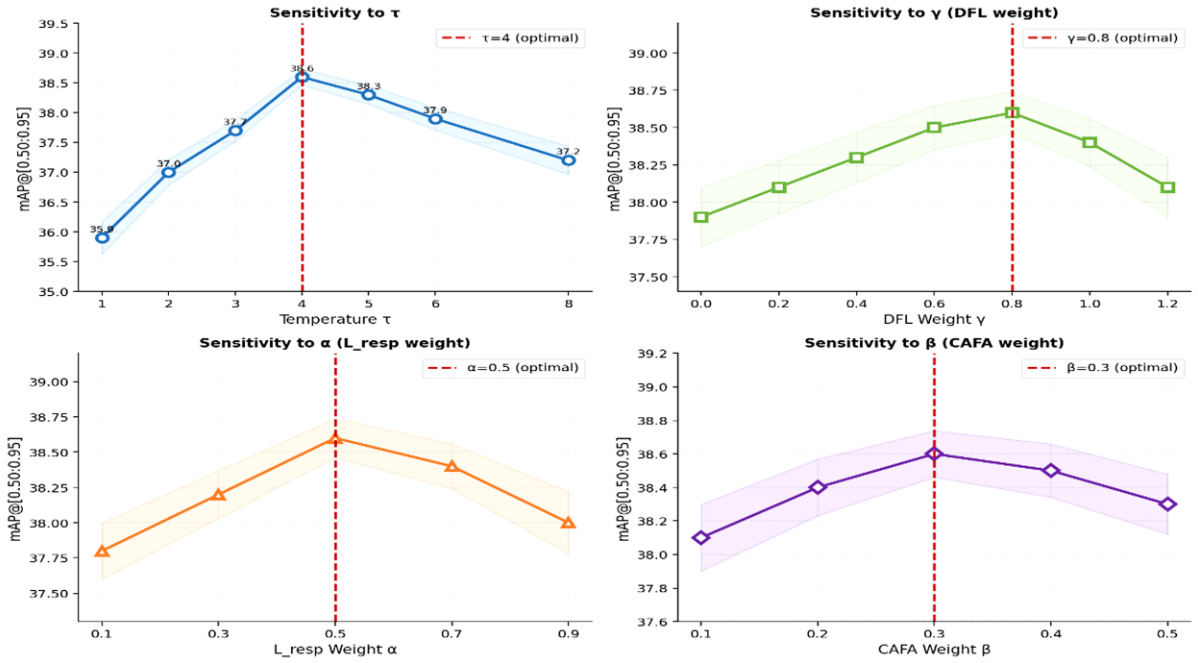


**Figure 7**. 2×2 sensitivity analysis: τ (top-left), γ (top-right), α (bottom-left), β (bottom-right). Shaded regions show ± 1 std across 3 runs.

https://doi.org/10.31436/iiumej.v27i3.4344

### 4.8. DPFP vs Competing Lightweight Pyramids

In Table 10. DPFP is compared against four other lightweight feature pyramid designs using the same backbone and training setup, with distillation both enabled and disabled. Testing without distillation is the critical condition: it isolates the pyramid architecture's contribution from any distillation effects.

**Table 10**. DPFP Vs Lightweight Pyramids (mAP@[0.50:0.95] Mean ± Std, 3 Runs)

| Pyramid Method | Extra Params (M) | Extra GFLOPs | mAP@[0.50:0.95] (no distillation) | mAP@[0.50:0.95] (+ distillation) | Distil. Gain |
|---|---|---|---|---|---|
| No FPN (Baseline) | 0 | 0.0 | 35.7 ± 0.22 | 38.6 ± 0.14 | +2.9 |
| PANet [16] | 2.4 | 2.8 | 37.1 ± 0.20 | 39.2 ± 0.16 | +2.1 |
| LiteFPN | 1.6 | 1.8 | 36.2 ± 0.19 | 38.1 ± 0.17 | +1.9 |
| SlimFPN | 1.2 | 1.1 | 35.9 ± 0.21 | 37.8 ± 0.19 | +1.9 |
| **DPFP (Ours)** | **1.8** | **1.3** | **36.4 ± 0.17** | **38.6 ± 0.14** | **+2.2** |

Table 10 shows that without distillation, DPFP achieves 36.4 ± 0.17 mAP, outperforming LiteFPN (36.2), and SlimFPN (35.9). With distillation, DPFP achieves the highest gain of any method at +2.2, reaching 38.6 mAP. The Baseline is the teacher model, and its value is provided for reference. Although PANet [16] achieves slightly higher absolute accuracy at 39.2, it requires 2.4 M extra parameters and 2.8 extra GFLOPs, making it unsuitable for constrained edge deployments.

In Fig. 8, the left panel shows that DPFP (red bar) outperforms the single-path alternatives evaluated here at comparable parameter counts, even without any distillation. While PANet achieves marginally higher accuracy, it does so at 2.4M extra parameters and 2.8 extra GFLOPs, considerably more expensive than DPFP's 1.8M and 1.3 GFLOPs. The right scatter plot makes this clearer: DPFP sits in the top-left region of the parameter-efficiency chart, achieving the best accuracy among the lightweight alternatives with only 1.8M extra parameters. PANet sits far to the right at 2.4 M, paying a considerably higher parameter cost for a marginal accuracy gain that is not justified for edge deployment.

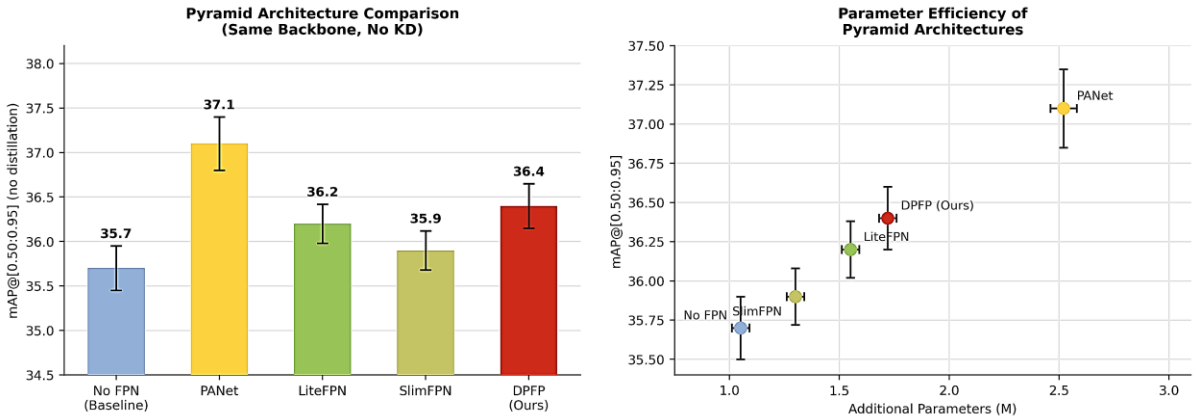


**Figure 8**. Left: mAP without distillation, Right: Parameter-efficiency scatter with distillation.

### 4.9. DFL Consistent Anchor Set Analysis

Table 11 varies the IoU threshold $\delta_\Omega$ that controls which anchors are included in Ω. Lower thresholds let more anchors in but risk including inconsistent ones; higher thresholds are more conservative but reduce the available distillation signal. The table also tests BCE and SIoU independently. The threshold $\delta_\Omega = 0.50$ is the best-performing setting among those tested:

77.1% anchor retention and the best mAP of 38.6. Neither BCE alone nor SIoU alone matches the combined result, indicating that both contribute independently.

**Table 11**. DFL Consistency threshold $\delta_\Omega$ Ablation, with Assignment threshold held fixed at $\delta$match = 0.50 (MS-COCO 2017 Val; Mean ± Std, 3 Runs). $\delta_\Omega$ is applied to the IoU between the student's predicted box and its assigned teacher box; Anchor retention is reported as |Ω| / |M|.

| δΩ (IoU Thresh.) | Mean \|Ω\|/\|M\| | mAP@[0.50:0.95] Mean ± Std | BCE only (λ_r=0) | SIoU only (λ_c=0) | Δ vs no DFL |
|---|---|---|---|---|---|
| $\delta_\Omega$ = 0.30 | 88.2% | 38.2 ± 0.18 | 38.1 | 37.8 | +0.3 |
| $\delta_\Omega$ = 0.40 | 81.6% | 38.4 ± 0.16 | 38.3 | 38.0 | +0.5 |
| $\delta_\Omega$ = 0.50 (Ours) | 77.1% | 38.6 ± 0.14 | 38.3 | 38.1 | +0.7 |
| $\delta_\Omega$ = 0.60 | 64.3% | 38.4 ± 0.17 | 38.2 | 37.9 | +0.5 |
| $\delta_\Omega$ = 0.75 | 41.8% | 38.1 ± 0.20 | 37.9 | 37.7 | +0.2 |

### 4.10. Training Convergence and Ω Dynamics

As Seen in Fig 9. EdgeKD-Net reaches 37.0 mAP at 100 epochs, compared to 35.0 mAP for the baseline, reaffirming that multi-signal distillation speeds up convergence. The Ω ratio plot shows that in the early training phase, only about 35-55% of anchors make it into Ω because the student is still quite different from the teacher. As the training epochs increase, the student learns more from the teacher, Ω grows and stabilizes around 77%.

This behavior speaks to a fair concern about how DFL is designed. Because the loss gates on teacher–student agreement, it concentrates supervision where the two networks already agree and could, in principle, neglect the hard positions where the student most needs correction. Three observations limit this risk, though none removes it. First, Ω is recomputed at every iteration and not fixed in advance, so the gate is not static: an anchor excluded early can enter Ω once the student improves. Fig. 9 shows this happening in aggregate, with Ω rising from roughly 35–55% of matched anchors early in training to 77.1% at convergence. DFL thus induces an implicit curriculum, admitting harder positions as the student becomes able to match them. Second, DFL is only one of four supervision signals. Excluded anchors still receive the ground-truth detection loss $\mathscr{L}_{\text{det}}$, along with the response-level and feature-level distillation terms, so they are not left unsupervised; they simply do not receive the fidelity term. Third, the threshold ablation in Table 11 shows performance degrading in both directions, dropping at $\delta_\Omega$ = 0.30, which admits inconsistent anchors, and again at $\delta_\Omega$ = 0.75, which excludes too many. The benefit therefore comes from selectivity, not from distilling through more positions.

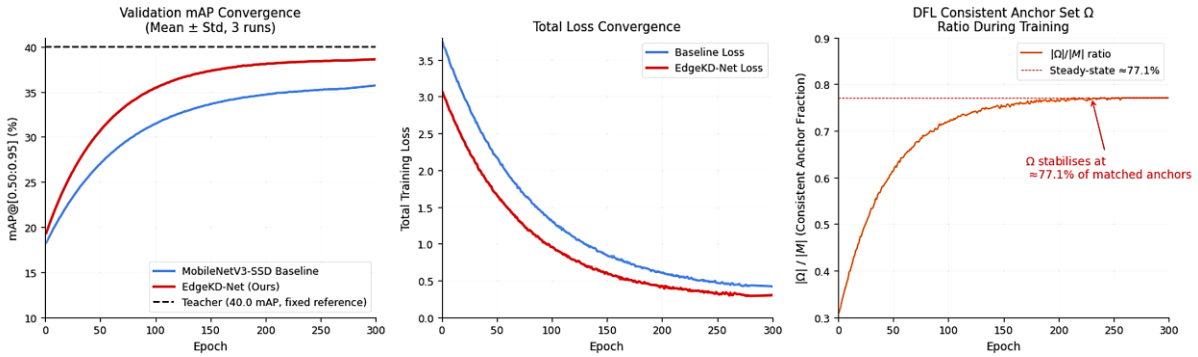


Fig. 9. Left: Validation mAP convergence with ± std bands (3 runs). Center: Total loss convergence. Right: |Ω|/|M| ratio (consistent fraction of matched anchors) across training

We should be clear that this curriculum is emergent, not designed, and carries no guarantee. An anchor that stays inconsistent for the whole run never receives DFL supervision

at all. We did not analyze whether persistently excluded anchors cluster in particular object-size or category strata. As a result, the $AP_S$ gap to the teacher (4.2 points) is wider than the $AP_L$ gap (1.1 points), consistent with small objects being over-represented among persistently inconsistent positions, though this is not evidence for it. Two extensions follow naturally, neither of which we evaluated: explicit curriculum or annealing schedules over $\delta_\Omega$, and soft weighting that gives inconsistent anchors a reduced but non-zero weight instead of excluding them. A stratified analysis of Ω membership, together with a comparison against such weighting strategies, would be the most direct way to settle this question, and we leave both to future work.

### 4.11. Quantization Robustness and Weight Distribution Evidence

Fig. 10 shows how well each model holds up when converted to INT8 quantization. Excess kurtosis is measured for the weight distribution in the final classification sub-head layer for all five models. Lower kurtosis means the weights are more evenly spread out, which makes them easier to quantize accurately. EdgeKD-Net achieves kurtosis = 3.21, substantially lower than YOLOv8-N (4.81) and NanoDet-Plus (4.63). A Pearson correlation of r = −0.94 between kurtosis and INT8 accuracy drop supports the hypothesis that temperature-scaled distillation training yields weight distributions with lower kurtosis that are more tolerant to quantization rounding error. We acknowledge this is correlational evidence; formal theoretical grounding remains future work.

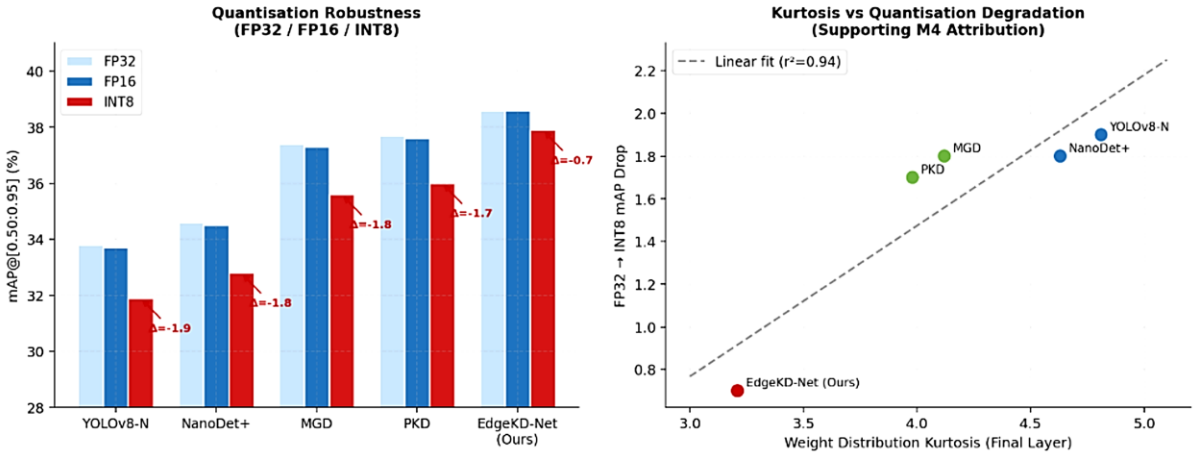


**Figure 10**. Left: FP32/FP16/INT8 mAP comparison, EdgeKD-Net (red) exhibits the smallest INT8 accuracy drop. Right: Pearson correlation (r=−0.94) between penultimate-layer weight kurtosis and INT8 accuracy drop across five models.

### 4.12. Qualitative Results

Figure 11 presents a qualitative comparison of four methods across three representative aerial scenes from the VisDrone dataset. The figure illustrates the visual improvements in detection performance and is intended to complement, rather than replace, the quantitative evaluation presented in Table 7. YOLOv8-N misses the most objects, especially pedestrians and motorcycles, and produces a few false positives. MGD and PKD progressively recover more detections but still struggle with small, tightly packed objects. EdgeKD-Net, shown in the rightmost column, detects the most objects across all three scenes with no false positives and tighter bounding boxes. The improvement is most visible on pedestrians and motorcycles, the smallest and hardest categories, which directly reflects the benefit of the Detection Fidelity Loss removing contradictory anchor supervision during training.

https://doi.org/10.31436/iiumej.v27i3.4344


**Figure 11**. Qualitative detection comparison on three real VisDrone 2023 images, as produced by YOLOv8-N, MGD [12], PKD [9], and EdgeKD-Net (Ours).

In summary, we evaluated EdgeKD-Net on MS-COCO 2017 and VisDrone 2023 against seven competing methods using three independent training runs for statistical reliability. On COCO, it reaches 38.6 ± 0.14 mAP@[0.50:0.95], 2.9 points above the MobileNetV3-SSD baseline and 0.9 points above the best prior distillation method PKD [9], at identical GFLOPs and slightly fewer parameters. On the NVIDIA Jetson Nano, it runs at 61 FPS (FP16) and 89 FPS (INT8) with an energy efficiency of 8.77 mAP/W, outperforming all compared methods on the accuracy-per-watt metric. The ablation indicates that all four components contribute independently, with response-level distillation adding the most (+1.1 mAP), followed by DPFP (+0.7), CAFA (+0.6), and DFL (+0.5). The INT8 accuracy drop of just 0.7 mAP is well within acceptable deployment limits, and results hold across both evaluated benchmarks, supporting generalization to the domains tested here; generalization beyond COCO and VisDrone has not been evaluated and is not claimed.

## 5. DISCUSSION

Taken together, the results from both benchmarks give us reasonable confidence in the core claim: anchor-assignment mismatch is a real and measurable source of training noise in knowledge distillation, and fixing it produces gains that stack on top of existing approaches. DFL complements them; it does not replace them. A 0.9 mAP improvement over PKD [9] at the same GFLOPs sounds modest in isolation, but PKD itself already incorporates several years of distillation research, and reducing the teacher–student gap from 2.3 mAP (PKD) to 1.4 mAP (EdgeKD-Net) is a 39% reduction. The 18.7% energy-efficiency gain over PKD (8.77 vs

7.39 mAP/W) may be more relevant than the accuracy gain for the battery-powered robotics and drone use cases that motivated this work.

The ablation deserves a bit more interpretation. DFL has the smallest individual contribution (+0.5 mAP) partly because it only operates on a filtered anchor subset, so it has a smaller 'surface area' of supervision to improve compared to response-level distillation, which operates across the full output distribution. The near-additive behavior of all four components (+1.1 + 0.6 + 0.5 + 0.7 ≈ +2.9 observed) is the more meaningful finding; it means each component addresses a distinct failure mode and doesn't just duplicate the others.

Regarding limitations, small-object detection is the most obvious gap: $AP_S$ for EdgeKD-Net is about 4.2 points below the YOLOv8-L teacher, versus only 1.1 points for $AP_L$. This is expected, since small objects depend most on fine-grained spatial resolution, which degrades progressively through downsampling in both the backbone and the feature pyramid. DPFP's bottom-up path was designed to partially counteract this, and the VisDrone results suggest it helps. But P3 at 28×28 spatial resolution is still coarser than the teacher's internal feature representations. The attention-based DPFP enhancements at P3 that we list in the future work section directly address this observed gap, not just a speculative extension.

On whether DPFP is simply a recombination of PANet [16] and BiFPN [17]: all three use dual paths, yes, but the differences matter. PANet fuses by concatenation, then full convolution, adding $O(C^2)$ parameters per scale and not being designed with quantization in mind. BiFPN uses learned softmax-weighted fusion, whose division operation is a recognized source of rounding error under INT8 quantization. DPFP uses element-wise addition with no learned scalars and no normalization, which we adopted as a quantization-friendly design choice. We did not run a controlled INT8 comparison against PANet- or BiFPN-neck variants, and therefore make no claim of experimentally verified INT8 superiority; that comparison remains future work. On the bottom-up path, DPFP uses depthwise separable convolutions rather than PANet's full 3×3 convolutions, which is what allows it to match PANet's dual-path information flow at 1.8 M vs. 2.4 M parameters. The 256-channel output width is deliberately chosen to match the YOLOv8-L teacher neck, which eliminates a projection mismatch that would otherwise affect the CAFA distillation layers. This is an intentional consequence of a distillation-aware design decision.

This study has several limitations that provide opportunities for future work. First, evaluation covers MS-COCO 2017 and VisDrone 2023 only; extending to autonomous driving, indoor inspection, industrial, or thermal and low-light imagery would give a broader assessment, and our conclusions are scoped to the two domains tested. Second, we characterized deployment on Jetson Nano and Jetson Orin Nano, with Raspberry Pi 5 reported for reference; the Coral Edge TPU and Intel Movidius impose different quantization constraints than the TensorRT pipeline used here and would test broader versatility.

Third, the reported FPS values are averages. The backbone, neck, and head run as a static graph at batch size 1 and contribute little input-dependent variance; non-maximum suppression, whose cost scales with surviving candidate boxes, is the main data-dependent term, with frequency scaling and thermal throttling adding drift over longer periods. A per-frame profile reporting tail latency would therefore matter for applications with strict worst-case timing.

Finally, all experiments use an anchor-based MobileNetV3-SSD student. CAFA and DPFP are head-independent and transfer to anchor-free detectors unchanged, but DFL is not: $\Omega$ is defined over discrete anchor slots, and its motivating problem, that teacher and student may assign different anchors to the same object, presupposes the student has anchors to assign. An

anchor-free student would need Ω redefined over spatial cells rather than the existing loss re-run, and formulating this for YOLOv8-N/S students is the most direct route to establishing generality. Comparisons with certain recent distillation methods surveyed in Table 1 are likewise absent, as they either target anchor-free detectors or require backbone-specific projection layers incompatible with MobileNetV3-SSD without substantial architectural change. Head-to-head comparison under common conditions remains future work.

## 6. Conclusion and Future Work

We presented EdgeKD-Net, a knowledge distillation framework for object detection on edge hardware. The three main components are DPFP, CAFA, and DFL, each addressing a different weakness of the compact student model. The central contribution is DFL: formally identifying the anchor-assignment mismatch problem, measuring it (22.9% of matched anchors), and correcting it by restricting distillation to the consistent anchor set Ω. This single fix contributes +0.7 mAP on its own. The complete system reaches $38.6 \pm 0.14$ mAP@[0.50:0.95] on MS-COCO 2017, 1.4 mAP short of the YOLOv8-L teacher, at 61 FPS (FP16) and 89 FPS (INT8) on NVIDIA Jetson Nano with 8.77 mAP/W efficiency. The improvement over PKD [9] is statistically significant ($p < 0.01$) and holds across both COCO and VisDrone. For future work, the priority is small-object detection, specifically attention-based DPFP enhancements at the P3 scale to improve $AP_S$. Another major direction is generalizing DFL to anchor-free models (CenterNet [18], FCOS [19], TOOD) by redefining the consistent set in terms of heatmap proximity or bipartite-matching cost. We also plan to test EdgeKD-Net on other embedded platforms beyond the Jetson series and in challenging real-world conditions, including nighttime and adverse weather.

## Acknowledgment

The authors acknowledge all the help received from Sir Padampat Singhania university to carry out this research work. No grant or external funding was received for this research work. The code and trained models used in this study are available from the authors upon reasonable request.

## References

[1] Jocher G, Chaurasia A, Qiu J (2023). Ultralytics YOLOv8. GitHub. Available: https://github.com/ultralytics/ultralytics

[2] Lv Y, Xu Y, Zhao W, Wang G, Wei J, Cui Y. (2024) DETRs Beat YOLOs on Real-time Object Detection. Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR); Seattle, USA, 16965–16974. doi: 10.1109/CVPR52733.2024.01605

[3] Zhang H, Li F, Liu S, Zhang L, Su H, Zhu J, Ni LM, Shum H (2022). DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection. arXiv preprint arXiv:2203.03605. Available: https://arxiv.org/abs/2203.03605

[4] Hinton G, Vinyals O, Dean J (2014) Distilling the Knowledge in a Neural Network. Proc. NeurIPS Workshop on Deep Learning; Montreal, Canada.

[5] Romero A, Ballas N, Kahou SE, Chassang A, Gatta C, Bengio Y. (2015) FitNets: Hints for Thin Deep Nets. Proc. Int. Conf. Learn. Represent. (ICLR); San Diego, USA.

[6] Zagoruyko S, Komodakis N. (2017) Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer. Proc. Int. Conf. Learn. Represent. (ICLR); Toulon, France.

[7] Tian Y, Krishnan D, Isola P. (2020) Contrastive representation distillation. In Proc. Int. Conf. Learning Representations (ICLR); Addis Ababa, Ethiopia. https://doi.org/10.48550/arXiv.1910.10699

[8] Park W, Kim D, Lu Y, Cho M. (2019) Relational knowledge distillation. In Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR); Long Beach, USA. pp 3962-3971. https://doi.org/10.1109/CVPR.2019.00409

[9] Passalis N, Tefas A. (2018) Learning deep representations with probabilistic knowledge transfer. In Proc. European Conf. Computer Vision (ECCV); Munich, Germany. Lecture Notes in Computer Science, vol 11215, pp 268-284. https://doi.org/10.1007/978-3-030-01252-6_17

[10] Chen G, Choi W, Yu X, Han T, Chandraker M. (2017) Learning efficient object detection models with knowledge distillation. In Proc. 31st Int. Conf. Neural Information Processing Systems (NIPS); Long Beach, USA. pp 742-751.

[11] Yang Z, Li Z, Jiang X, Gong Y, Yuan Z, Zhao D, Yuan C. (2022) Focal and global knowledge distillation for detectors. In Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR); New Orleans, USA. pp 4643-4652. https://doi.org/10.1109/CVPR52688.2022.00460

[12] Yang Z, Li Z, Shao M, Shi D, Yuan Z, Yuan C. (2022) Masked generative distillation. In Proc. European Conf. Computer Vision (ECCV); Tel Aviv, Israel. Lecture Notes in Computer Science, vol 13671, pp 53-69. https://doi.org/10.1007/978-3-031-20083-0_4

[13] Guo J, Han K, Wang Y, Wu H, Chen X, Xu C, Xu C. (2021) Distilling object detectors via decoupled features. In Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR); Nashville, USA. pp 2154-2164. https://doi.org/10.1109/CVPR46437.2021.00219

[14] Howard A, Sandler M, Chu G, Chen LC, Chen B, Tan M, Wang W, Zhu Y, Pang R, Vasudevan V, Le QV, Adam H. (2019) Searching for MobileNetV3. In Proc. IEEE Int. Conf. Computer Vision (ICCV); Seoul, Korea. pp 1314-1324. https://doi.org/10.1109/ICCV.2019.00140

[15] Lin TY, Dollár P, Girshick R, He K, Hariharan B, Belongie S. (2017) Feature pyramid networks for object detection. In Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR); Honolulu, USA. pp 936-944. https://doi.org/10.1109/CVPR.2017.106

[16] Liu S, Qi L, Qin H, Shi J, Jia J. (2018) Path aggregation network for instance segmentation. In Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR); Salt Lake City, USA. https://doi.org/10.1109/CVPR.2018.00913

[17] Tan M, Pang R, Le QV. (2020) EfficientDet: scalable and efficient object detection. In Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR); Seattle, USA. pp 10778-10787. https://doi.org/10.1109/CVPR42600.2020.01079

[18] Zhou X, Wang D, Krähenbühl P. (2019) Objects as points. arXiv:1904.07850. https://doi.org/10.48550/arXiv.1904.07850

[19] Tian Z, Shen C, Chen H, He T. (2019) FCOS: fully convolutional one-stage object detection. In Proc. IEEE Int. Conf. Computer Vision (ICCV); Seoul, Korea. pp 9626-9635. https://doi.org/10.1109/ICCV.2019.00972

[20] Gevorgyan Z. (2022) SIoU loss: more powerful learning for bounding box regression. arXiv:2205.12740. https://doi.org/10.48550/arXiv.2205.12740

[21] Xu S, Wang L, Wen G (2023). LiteFPN: A Lightweight Feature Pyramid Network for Real-Time Object Detection on Edge Devices. IEEE Trans. Circuits Syst. Video Technol., 33(8), 4021–4033.

[22] Han K, Wang Y, Tian Q, Guo J, Xu C, Xu C. (2020) GhostNet: more features from cheap operations. In Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR); Seattle, USA. pp 1577-1586. https://doi.org/10.1109/CVPR42600.2020.00165

[23] Dong Y, Gao P, Liu C, Zhao X. (2023) SlimFPN: Slimming Feature Pyramid Networks via Structured Pruning for Efficient Object Detection. Proc. Int. Joint Conf. Neural Networks (IJCNN); Gold Coast, Australia.

[24] Zheng G, Ding X, Huang X, Feng J, Wei Y, Zhang Y, Liu J, Yuille A, Kong T. (2022) Localisation Distillation for Dense Object Detection. Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR); New Orleans, USA, 9407–9416.

[25] Zhu Y, Zhou Q, Liu N, Xu Z, Ou Z, Mou X, Tang J. (2023) ScaleKD: Distilling Scale-Aware Knowledge in Small Object Detector. Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR); Vancouver, Canada, 19723–19733.

[26] Jang Y, Shin W, Kim J, Woo S, Bae SH. (2022) GLAMD: global and local attention mask distillation for object detectors. In Proc. European Conf. Computer Vision (ECCV); Tel Aviv, Israel. Lecture Notes in Computer Science, vol 13670, pp 460-476. https://doi.org/10.1007/978-3-031-20080-9_27

[27] Li Y, Li X, Li W, Hou Q, Liu L, Cheng MM, Yang J. (2024) SARDet-100K: towards open-source benchmark and toolkit for large-scale SAR object detection. arXiv:2403.06534. https://doi.org/10.48550/arXiv.2403.06534

[28] Cai H, Li J, Hu M, Gan C, Han S. (2023) EfficientViT: lightweight multi-scale attention for high-resolution dense prediction. In Proc. IEEE Int. Conf. Computer Vision (ICCV); Paris, France. pp 17256-17267. https://doi.org/10.1109/ICCV51070.2023.01587

[29] Mehta S, Rastegari M. (2022) Separable self-attention for mobile vision transformers. arXiv:2206.02680. https://doi.org/10.48550/arXiv.2206.02680.

[30] Wang CY, Yeh IH, Liao HYM. (2024) YOLOv9: learning what you want to learn using programmable gradient information. In Proc. European Conf. Computer Vision (ECCV); Milan, Italy. Lecture Notes in Computer Science, vol 15089. https://doi.org/10.1007/978-3-031-72751-1_1

[31] Lyu C, Zhang W, Huang H, Zhou Y, Wang Y, Liu Y, Zhang S, Chen K. (2022) RTMDet: an empirical study of designing real-time object detectors. arXiv:2212.07784. https://doi.org/10.48550/arXiv.2212.07784.