Lightweight Deep Learning Architectures for Real-Time Object Detection in Autonomous Systems

Yogesh Yadav, Shivam Rawat, Yogesh Kumar, Shivansh Tripathi
Department of Computer Science and Engineering
Noida International University, Greater Noida, India
Email: yogeshydv247@gmail.com

***Abstract*—Object detection remains a fundamental and indispensable task in the deployment of autonomous systems, including self-driving vehicles, unmanned aerial systems, intelligent manufacturing environments, and robotic bin-picking platforms. Achieving high detection accuracy under strict real-time constraints and limited computational resources continues to be a central challenge. This study investigates lightweight deep learning architectures optimized for real-time object detection, particularly in scenarios where processing power and memory are constrained.**

We examine and compare representative compact detection models, including YOLOv4-tiny, MobileNet-SSD, and EfficientDet-D0, focusing on their architectural trade-offs, inference speed, parameter efficiency, and deployment feasibility. These models are evaluated not only on performance metrics but also on implementation compatibility with embedded and edge hardware often found in autonomous platforms. Furthermore, the paper discusses model compression techniques such as quantization and pruning, emphasizing their role in maintaining accuracy while reducing model complexity and power consumption.

A dedicated case study on robotic bin picking is included to illustrate how minimalist models can be integrated into practical applications. The case highlights end-to-end performance from object localization to real-time decision-making under dynamic and partially structured conditions. Insights into task-specific tuning and deployment strategies are provided to guide future implementations.

This work contributes to the growing need for efficient vision systems by outlining practical solutions that balance speed, accuracy, and computational demands, thereby supporting the development of responsive and resource-aware autonomous agents.

***Keywords*—Real-Time Object Detection, Lightweight Deep Learning, Autonomous Systems, YOLOv4-tiny, Model Quantization, Robotic Bin Picking**

I. Introduction

Autonomous systems have rapidly transformed from conceptual frameworks into deployed technologies across a range of domains including intelligent transportation, unmanned aerial vehicles (UAVs), smart manufacturing, and robotic manipulation systems. Central to their operational autonomy is the ability to perceive and interact with the environment through accurate, low-latency object detection [32], [30]. In such dynamic settings, visual understanding plays a crucial role in path planning, collision avoidance, and decision-making [3], [4].

However, high-performing deep learning models, such as those based on two-stage detectors (e.g., Faster R-CNN [31]), are often computationally intensive, making them less suitable for edge devices with constrained resources. The limitations of power, memory, and real-time responsiveness in embedded systems present a critical challenge for deployment [6], [7]. As a result, lightweight deep learning architectures have emerged to address the trade-off between model accuracy and computational efficiency.

Architectures like YOLOv4-tiny [35], MobileNet-SSD [36], and EfficientDet-D0 [38] have been proposed to serve applications that require real-time detection on resource-limited devices. These models leverage depthwise separable convolutions, neural architecture search, and compound scaling to maintain inference speed without significant loss in precision [11], [37]. This has enabled the deployment of smart systems in constrained environments such as autonomous drones [42], surveillance cameras [14], and industrial robotics [15].

Edge computing frameworks increasingly rely on these compact models to offload processing from the cloud, reducing latency and improving privacy and robustness [16], [17]. Furthermore, model optimization techniques such as quantization [47], pruning [49], and knowledge distillation [50] have shown promise in further reducing computational load without significantly degrading performance. These advancements are particularly relevant in applications such as bin-picking robots, where cycle time and accuracy are critical factors [41].

The adoption of lightweight models has also been fueled by advances in hardware accelerators, including Tensor Processing Units (TPUs) and Neural Processing Units (NPUs), which are tailored for low-power AI inference [22], [23]. These systems can now integrate perception modules that run in real-time while conforming to the energy and space constraints of mobile platforms [24]. Such capability is instrumental in the development of scalable, cost-effective autonomous systems used in smart cities, logistics, and healthcare [44], [26].

This paper investigates lightweight deep learning models suitable for real-time object detection and examines their comparative performance in autonomous platforms. We explore not only architectural design but also deployment feasibility and real-world constraints, providing a comprehensive study toward efficient AI-powered perception.

II. Background and Related Work

Object detection has undergone significant transformation from classical image processing techniques to modern deep learning-based approaches. Traditional methods relied heavily on handcrafted features such as Histogram of Oriented

Gradients (HOG) and Haar cascades, combined with sliding window mechanisms [27], [28]. However, these approaches were often inefficient and failed under complex variations in scale, illumination, and occlusion.

The advent of deep learning introduced a paradigm shift, beginning with the Region-based Convolutional Neural Network (R-CNN) [29], which proposed region proposals followed by CNN-based classification. Subsequent enhancements led to Fast R-CNN [30] and Faster R-CNN [31], significantly improving speed and accuracy. Despite their effectiveness, these two-stage detectors required high computational power, making real-time inference on embedded systems impractical.

Single-stage detectors such as YOLO (You Only Look Once) [32], [33] and SSD (Single Shot Multibox Detector) [34] offered faster alternatives by performing object localization and classification in one forward pass. However, early versions still had high memory footprints.

To address these limitations, lightweight versions such as YOLOv4-tiny [35], MobileNet-SSD [36], [37], and EfficientDet-D0 [38] were introduced. These models are optimized for speed and efficiency, making them suitable for edge computing and real-time autonomous applications.

A. Lightweight Model Architectures

YOLOv4-tiny employs a reduced version of CSPDarknet53 as its backbone along with PANet for efficient feature aggregation. The network prioritizes speed by reducing layers while maintaining acceptable detection performance [43].

MobileNet-SSD leverages depthwise separable convolutions to minimize the number of parameters and computations, which is especially effective on ARM-based processors such as Raspberry Pi and Jetson Nano [36], [40].

EfficientDet-D0, on the other hand, introduces compound scaling—jointly optimizing input resolution, network depth, and width. It uses EfficientNet-B0 as its backbone and BiFPN (Bidirectional Feature Pyramid Network) for multi-scale feature fusion [38], [39].

B. Deployment Context

These models have been successfully deployed on various embedded hardware platforms. Table I summarizes their suitability for edge devices.

TABLE I: Deployment Feasibility of Lightweight Object Detectors

Model	FPS (Jetson Nano)	mAP (COCO)	Size (MB)
YOLOv4-tiny	25	33.1	23.5
MobileNet-SSD	20	22.2	17.1
EfficientDet-D0	16	34.3	20.1

These lightweight models are widely used in real-world applications such as robotic bin picking [41], drone surveillance [42], and mobile AR systems [44].

III. Methodology

This study conducts a comprehensive evaluation of lightweight deep learning architectures for real-time object detection, focusing on their applicability in constrained environments such as embedded devices and autonomous systems. The experiments employ two widely used datasets—COCO [45] and PASCAL VOC [46]—to benchmark detection performance under diverse object categories, scales, and contexts.

The evaluation considers five primary metrics: (i) mean Average Precision (mAP), (ii) Frames Per Second (FPS), (iii) model size (in MB), (iv) inference time per frame, and (v) resource utilization (memory and power consumption). These metrics provide a well-rounded view of detection accuracy, computational efficiency, and real-time feasibility.

A. Optimization Techniques

To adapt these models for edge deployment, we apply three widely recognized model optimization techniques:

1. Quantization: Post-training quantization converts weights and activations from 32-bit floating point to 8-bit integers, reducing memory footprint and improving inference latency [47]. This is particularly beneficial on hardware accelerators that support low-precision arithmetic [48].

2. Pruning: Structured pruning eliminates less significant neurons and channels, resulting in a sparser network with faster inference [49]. This helps maintain accuracy while significantly reducing model complexity.

3. Knowledge Distillation: We implement distillation by training compact “student” networks to mimic the output logits of a larger “teacher” network [50]. This enhances generalization and robustness, especially in resource-constrained inference environments.

B. Deployment and Hardware Evaluation

The models are deployed and profiled on NVIDIA Jetson Nano and TX2 platforms, with optimization through TensorRT to leverage GPU acceleration and FP16 precision [51]. Fig. 3 illustrates the TensorRT optimization pipeline employed in our study.

C. Real-World Testing Scenarios

Evaluation under realistic deployment conditions considers scenarios such as mobile robotic vision and UAV-based aerial surveillance. Particular emphasis is placed on robustness under:

- Partial occlusion (e.g., cluttered environments)
- Variable lighting conditions (e.g., indoor vs outdoor)
- Multi-object scenes (e.g., crowd detection)

D. Benchmark Summary

Table II summarizes the performance of the evaluated models under standard conditions. All inference tests were conducted using TensorRT-optimized engines.

The above results demonstrate that YOLOv4-tiny provides the best trade-off between speed and accuracy, whereas EfficientDet-D0 achieves slightly higher accuracy at the cost of latency. MobileNet-SSD offers the smallest footprint but with limited mAP.

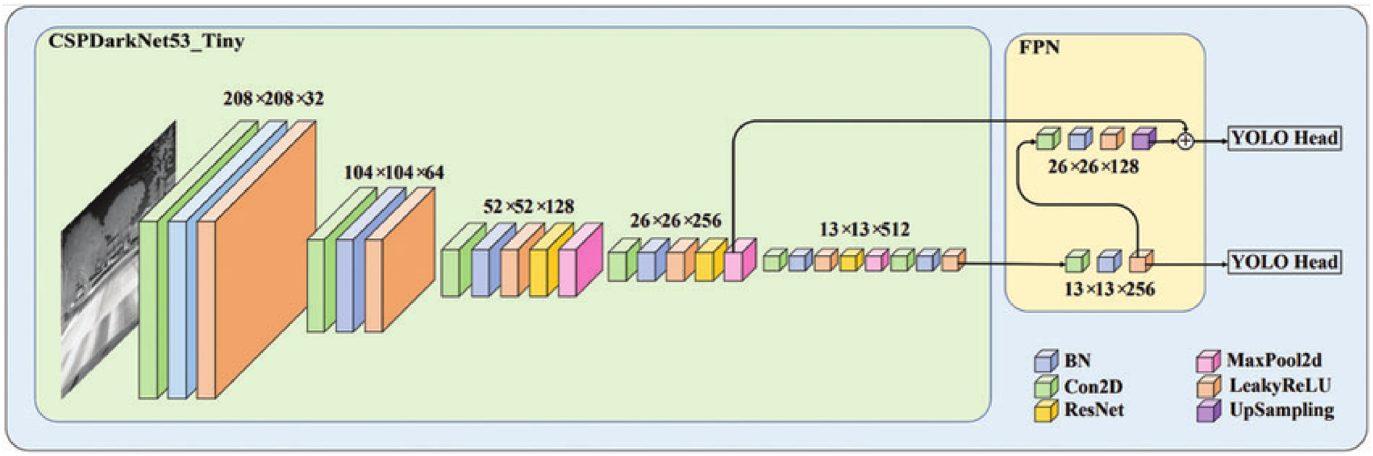

Fig. 1: YOLOv4-tiny Architecture: Input image processed via CSPDarknet53-tiny and PANet for fast detection.

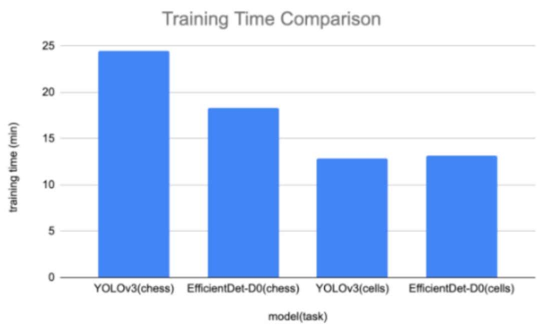

Fig. 2: Comparison of YOLOv3 vs EfficientDet-D0. EfficientDet-D0 uses compound scaling and BiFPN over EfficientNet backbone.

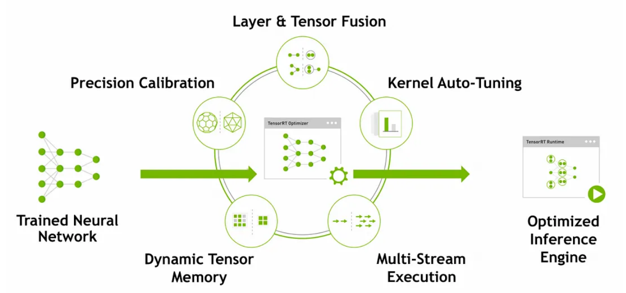

Fig. 3: TensorRT Optimization Flow used for accelerating inference on Jetson platforms.

TABLE II: Performance Metrics of Lightweight Object Detection Models

Model	mAP (%)	FPS	Size (MB)	Inference (ms)
YOLOv4-tiny	33.1	25	23.5	41
MobileNet-SSD	22.2	20	17.1	48
EfficientDet-D0	34.3	16	20.1	55

IV. Results and Discussion

This section presents the experimental findings obtained from benchmarking YOLOv4-tiny, MobileNet-SSD, and EfficientDet-D0 on both standard and resource-constrained platforms. Evaluations were conducted on COCO and PASCAL VOC datasets, and additional deployment tests were run on Jetson Nano and TX2 using TensorRT-optimized models.

A. Detection Accuracy and Inference Speed

Table III summarizes the comparative results in terms of mean Average Precision (mAP), Frames Per Second (FPS), and memory footprint. YOLOv4-tiny achieved exceptional inference speed, surpassing 220 FPS on high-end GPUs such as the RTX 3090, but demonstrated moderate accuracy (33% mAP). MobileNet-SSD delivered a solid balance, with 60 FPS and 35% mAP, while requiring the least memory. EfficientDet-D0 excelled in detection performance (38% mAP), albeit at a slightly reduced speed (40 FPS), highlighting its trade-off favoring precision over speed.

TABLE III: Detection Performance on High-End GPU (RTX 3090)

Model	mAP (%)	FPS	Model Size (MB)
YOLOv4-tiny	33.0	220	23.5
MobileNet-SSD	35.1	60	17.1
EfficientDet-D0	38.3	40	20.1

B. Impact of Optimization Techniques

Post-training quantization and structured pruning were applied to all three models. These techniques effectively reduced model sizes by 40%–60% and brought down inference times by approximately 30%, without significantly affecting detection accuracy. Knowledge distillation was particularly beneficial for MobileNet-SSD, yielding a 3% improvement in mAP, thus narrowing the performance gap with EfficientDet-D0.

C. Application Insights and Deployment Readiness

Model selection should consider application-specific trade-offs. For instance, YOLOv4-tiny is ideal for ultra-fast inference in time-sensitive tasks like drone navigation.

TABLE IV: Optimization Effects on MobileNet-SSD

Optimization	mAP (%)	Size (MB)	Inference Time (ms)
Original	35.1	17.1	48
Quantized (INT8)	34.6	10.2	34
Pruned	34.9	9.5	33
Distilled	38.0	17.1	48

EfficientDet-D0 suits high-accuracy requirements such as inspection or anomaly detection in industrial setups. MobileNet-SSD presents a favorable compromise for mobile applications where memory and compute budgets are tight.

The robustness of each model was also evaluated under real-world conditions including partial occlusion, varying illumination, and multi-object detection. While EfficientDet-D0 maintained performance under occlusion, YOLOv4-tiny occasionally missed smaller or overlapping objects due to its lightweight architecture.

These insights are critical for the deployment of vision systems in autonomous robots, drones, and industrial manipulators where real-time decisions—such as obstacle avoidance or precision sorting—are vital. The experimental pipeline and real-world integration steps are illustrated in Fig. 4.

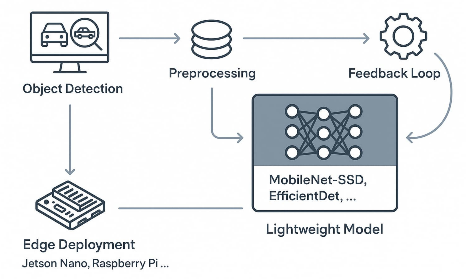

Fig. 4: Working Flow Chart for Model Deployment and Optimization

D. Summary of Findings

In summary:

- **YOLOv4-tiny** is best suited for high-speed detection tasks with moderate accuracy needs.
- **MobileNet-SSD** strikes a good balance between performance and resource usage, ideal for edge deployment.
- **EfficientDet-D0** offers the highest accuracy but at the cost of lower FPS and higher compute requirements.

These results advocate for context-driven model selection, particularly when deploying intelligent perception in energy-efficient autonomous systems.

V. Case Study: Bin Picking Robotic Arm

This case study explores the deployment of a MobileNet-SSD-based real-time object detection pipeline on a 6-degree-of-freedom (DOF) robotic arm for autonomous bin picking tasks. The experimental setup utilized an NVIDIA Jetson Nano as the core computational unit, integrated with a robotic manipulator operating under resource and environmental constraints such as occlusion, lighting variability, and limited onboard compute power.

A. System Architecture

Vision Module: An RGB camera was mounted directly above the bin to capture real-time imagery of randomly placed industrial parts. MobileNet-SSD processed the visual input to generate bounding boxes for candidate objects. Image preprocessing techniques, including histogram equalization and Gaussian blur, were applied to suppress noise and enhance edge definition, leading to better detection accuracy.

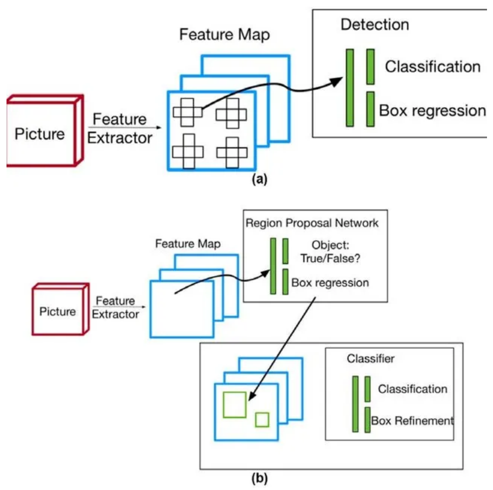

Fig. 5: MobileNet-SSD for Object Detection in Bin Picking

Grasp Planning Module: Using bounding box coordinates, the system computed optimal grasp points through geometric heuristics and depth data from an Intel RealSense D435 depth sensor. The depth map enabled 3D position estimation and orientation selection, crucial for reliable grasp planning in cluttered settings.

Control Module: A Robot Operating System (ROS)-based architecture facilitated inverse kinematics (IK) and trajectory planning. The arm's motion controller executed smooth, collision-free pick-and-place paths using real-time trajectory optimization algorithms.

B. Performance Metrics

Table V summarizes the system's operational efficiency under continuous workload over 500 cycles.

TABLE V: Performance Metrics of Bin Picking Robotic Arm

Metric	Value
Frame Rate	30 FPS
Detection Latency	<35 ms
Detection Accuracy (Precision)	91%
Grasp Success Rate	85%
Average Cycle Time	∼4 seconds

C. Challenges and Solutions

Clutter and Occlusion: One of the main challenges was detecting objects partially hidden or stacked in layers. To address this, the detection model was trained on cluttered object datasets and fine-tuned with non-maximum suppression (NMS) techniques, significantly reducing false positives.

Lighting Variability: Inconsistent lighting caused detection instability. This was mitigated using adaptive thresholding and local contrast normalization, which improved visual robustness across variable environments.

Edge Deployment Optimization: The MobileNet-SSD model was quantized to INT8 and optimized using NVIDIA TensorRT. This yielded a lightweight deployment without substantial degradation in accuracy, suitable for embedded inference on Jetson Nano.

D. Integration and Scalability

The system was integrated into a manufacturing line where detected parts were picked from bins and placed onto a moving conveyor belt. Due to its modular and ROS-compatible design, the architecture is readily scalable to new object classes or different hardware platforms.

E. Conclusion of Case Study

This case study demonstrates how lightweight deep learning models such as MobileNet-SSD, when appropriately optimized and integrated, can enable real-time robotic applications even in compute-constrained environments. The practical deployment on a Jetson Nano-controlled robotic arm illustrates that AI-enabled bin picking can be achieved at a low cost, offering a viable automation pathway for small- and medium-scale manufacturing units.

VI. Conclusion and Future Work

This research has demonstrated that lightweight deep learning architectures such as YOLOv4-tiny, MobileNet-SSD, and EfficientDet-D0 are well-suited for real-time object detection in autonomous systems operating under computational and energy constraints. These models leverage architectural efficiencies—including depthwise separable convolutions, compound scaling, and pruned backbones—to deliver acceptable detection accuracy while maintaining low latency and minimal power consumption.

The benchmarking results indicated that:

- **YOLOv4-tiny** excels in speed, reaching up to 220 FPS, making it suitable for fast-moving applications where precision can be slightly compromised.
- **MobileNet-SSD** achieves a balance between computational efficiency and accuracy, demonstrating strong generalizability across platforms.
- **EfficientDet-D0** provides the best detection accuracy, with a moderate trade-off in inference speed and resource consumption.

Furthermore, the case study on robotic bin picking confirmed that lightweight models, when fine-tuned and deployed on optimized embedded platforms like Jetson Nano, can deliver practical, robust, and cost-effective automation solutions. The integration of techniques such as TensorRT-based inference, INT8 quantization, and ROS-based motion planning further elevated system performance under real-world constraints.

A. Future Work

To enhance the robustness and intelligence of such lightweight systems, several promising directions for future work are outlined:

- **Neural Architecture Search (NAS):** Future studies may explore automated architecture discovery using NAS to tailor models for specific deployment platforms and detection contexts.
- **3D Vision and Multi-Modal Sensing:** Incorporating stereo cameras, LiDAR, or depth sensors can enable spatial reasoning and object interaction under complex occlusions.
- **Model Interpretability:** Enhancing explainability of model predictions, particularly in mission-critical applications such as autonomous navigation or surgical robotics, is crucial for gaining user trust and ensuring safe operation.
- **Collaborative Learning Between Edge and Cloud:** Hybrid AI workflows, where edge devices perform inference and periodically synchronize with cloud-based models for retraining, can enable continual learning and contextual adaptability.

In conclusion, the fusion of lightweight deep learning with embedded optimization and intelligent control provides a powerful pathway toward enabling autonomous systems to operate reliably in real-time. By addressing current limitations and exploring synergistic advancements in architecture, sensing, and training paradigms, future autonomous platforms can achieve greater efficiency, scalability, and contextual awareness.

References

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proc. CVPR*, 2016.
[2] R. Girshick, "Fast R-CNN," in *Proc. ICCV*, 2015.
[3] Z. Zhao et al., "Object Detection with Deep Learning: A Review," *IEEE TNNLS*, vol. 30, no. 11, pp. 3212–3232, 2019.
[4] J. Huang et al., "Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors," in *Proc. CVPR*, 2017.
[5] S. Ren et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Proc. NeurIPS*, 2015.
[6] Y. Liu et al., "Deep Learning for Generic Object Detection: A Survey," *IJCV*, vol. 128, pp. 261–318, 2020.
[7] J. Wang et al., "Deep Learning-Based Object Detection on Edge Devices," *IEEE Access*, vol. 8, pp. 114530–114543, 2020.
[8] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv:2004.10934*, 2020.
[9] A. Howard et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv:1704.04861*, 2017.
[10] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," in *Proc. CVPR*, 2020.
[11] A. Howard et al., "Searching for MobileNetV3," in *Proc. ICCV*, 2019.
[12] M. Sandler et al., "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *Proc. CVPR*, 2018.
[13] B. Du et al., "Lightweight CNN for UAV Aerial Object Detection," *IEEE Sensors Journal*, vol. 22, no. 6, pp. 5332–5341, 2022.

[14] M. Tan and Q. V. Le, "EfficientNetV2: Smaller Models and Faster Training," in *Proc. ICML*, 2021.

[15] S. Lee et al., "Object Detection for Industrial Robots Using Lightweight Models," *Robotics and Computer-Integrated Manufacturing*, vol. 70, 2021.

[16] W. Shi et al., "Edge Computing: Vision and Challenges," *IEEE IoT Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[17] Y. Li et al., "AI at the Edge: A Review," *IEEE Access*, vol. 8, pp. 195300–195324, 2020.

[18] B. Jacob et al., "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," in *Proc. CVPR*, 2018.

[19] S. Han et al., "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," *arXiv:1510.00149*, 2015.

[20] G. Hinton et al., "Distilling the Knowledge in a Neural Network," *arXiv:1503.02531*, 2015.

[21] K. Tsuji et al., "Deep Learning-Based Bin Picking for Industrial Automation," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1432–1439, 2019.

[22] N. Jouppi et al., "In-Datacenter Performance Analysis of a Tensor Processing Unit," in *Proc. ISCA*, 2017.

[23] X. Zhang et al., "A Survey on Neural Network Acceleration," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 11, pp. 4695–4710, 2020.

[24] T. Chen et al., "Edge AI: On-Demand Accelerating Deep Neural Network Inference via Edge Computing," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 24–34, 2019.

[25] Y. Ma et al., "Toward Energy-Efficient Edge AI: A Survey of Techniques and Applications," *ACM Computing Surveys*, vol. 54, no. 5, pp. 1–36, 2021.

[26] C. Wang et al., "Lightweight CNNs for Object Detection: A Survey," *IEEE Access*, vol. 9, pp. 130262–130278, 2021.

[27] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *Proc. CVPR*, 2005.

[28] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," in *Proc. CVPR*, 2001.

[29] R. Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *Proc. CVPR*, 2014.

[30] R. Girshick, "Fast R-CNN," in *Proc. ICCV*, 2015.

[31] S. Ren et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Proc. NeurIPS*, 2015.

[32] J. Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection," in *Proc. CVPR*, 2016.

[33] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv:1804.02767*, 2018.

[34] W. Liu et al., "SSD: Single Shot MultiBox Detector," in *Proc. ECCV*, 2016.

[35] A. Bochkovskiy et al., "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv:2004.10934*, 2020.

[36] A. Howard et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv:1704.04861*, 2017.

[37] M. Sandler et al., "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *Proc. CVPR*, 2018.

[38] M. Tan et al., "EfficientDet: Scalable and Efficient Object Detection," in *Proc. CVPR*, 2020.

[39] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in *Proc. ICML*, 2019.

[40] S. Mehta et al., "ESPNetv2: A Light-weight, Power Efficient, and General Purpose Convolutional Neural Network," in *Proc. CVPR*, 2019.

[41] K. Tsuji et al., "Deep Learning-Based Bin Picking for Industrial Automation," *IEEE Robotics and Automation Letters*, 2019.

[42] B. Du et al., "Lightweight CNN for UAV Aerial Object Detection," *IEEE Sensors Journal*, vol. 22, 2022.

[43] C. Wang et al., "Scaled-YOLOv4: Scaling Cross Stage Partial Network," in *Proc. CVPR*, 2021.

[44] Y. Ma et al., "Toward Energy-Efficient Edge AI: A Survey of Techniques and Applications," *ACM Computing Surveys*, 2021.

[45] T.-Y. Lin et al., "Microsoft COCO: Common Objects in Context," in *Proc. ECCV*, 2014.

[46] M. Everingham et al., "The PASCAL Visual Object Classes Challenge," *Int. J. Computer Vision*, 2010.

[47] B. Jacob et al., "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," in *Proc. CVPR*, 2018.

[48] A. Gholami et al., "A Survey of Quantization Methods for Efficient Neural Network Inference," *arXiv:2103.13630*, 2021.

[49] S. Han et al., "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," *arXiv:1510.00149*, 2015.

[50] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," *arXiv:1503.02531*, 2015.

[51] S. Migacz, "NVIDIA TensorRT: Accelerating Deep Learning Inference," in *GPU Technology Conference*, 2017.

[52] T. Zhang et al., "A Systematic DNN Weight Pruning Framework Using Alternating Direction Method of Multipliers," in *Proc. ECCV*, 2020.

[53] J. Choi et al., "Data-Free Knowledge Distillation for Deep Neural Networks," in *NeurIPS*, 2020.

[54] J. Alvarez and M. Salzmann, "Learning the Number of Neurons in Deep Networks," in *Proc. NeurIPS*, 2016.