Multi-Class Object Detection Using Quantized YOLOv11 for Real-Time Inference

Yehia A. Soliman[1], Amr Ghoneim[2], Mahmoud Elkhouly[3]
Faculty of Computers and Artificial Intelligence-Information Technology Department, Helwan University, Cairo, Egypt[1, 3]
Faculty of Computers and Artificial Intelligence-Computer Science Department, Helwan University, Cairo, Egypt[2]

***Abstract*—Real-time multi-class object detection on embedded devices poses significant challenges due to limited computational power, memory capacity, and energy efficiency requirements. Conventional high-precision object detectors, such as YOLOv11, deliver outstanding accuracy but are computationally intensive, making them unsuitable for deployment on resource-constrained hardware. This study presents a quantized implementation of the YOLOv11 model designed to enable efficient real-time inference on embedded platforms. The proposed approach applies post-training integer quantization and mixed-precision optimization to minimize computation and memory usage while maintaining detection accuracy across multiple object categories. Experimental evaluations were conducted on the COCO and Pascal VOC datasets. The results indicate that the quantized YOLOv11 achieves a 3.2× increase in inference speed, a 2.7× reduction in memory footprint, and a 35% improvement in energy efficiency, with less than 2% loss in mean Average Precision (mAP) compared to the full-precision baseline. The optimized model sustains real-time performance exceeding 45 frames per second (FPS), demonstrating that quantization is a viable and effective approach for deploying high-performance object detection models on embedded systems.**

Keywords*—Quantized neural networks; YOLOv11; object detection; embedded systems; real-time inference; model optimization

I. Introduction

The growing integration of intelligent vision systems into autonomous drones, surveillance networks, and Internet of Things (IoT) applications has increased the demand for real-time on-device object detection [1, 2]. Deploying inference directly on embedded devices reduces dependence on cloud infrastructure, thereby lowering latency, improving data privacy, and enabling continuous operation in bandwidth-limited environments. However, these devices typically feature limited computational resources, constrained memory, and strict power budgets. Traditional high-precision deep learning models, though highly accurate, are unsuitable for such hardware due to their intensive floating-point computations and large model sizes [3, 4]. Therefore, developing efficient and compact neural networks capable of real-time performance remains a significant research challenge in embedded artificial intelligence.

Object detection plays a vital role in computer vision tasks such as autonomous navigation, industrial inspection, and human-computer interaction [5]. Among the most prominent detectors, the You Only Look Once (YOLO) family has evolved rapidly from YOLOv1 to YOLOv11, introducing architectural innovations that enhance accuracy while maintaining high inference speed [6, 7, 8]. YOLOv11, the latest in this series, integrates improved feature fusion and efficient attention mechanisms for multi-scale detection. Despite these advances, high-precision YOLO models are computationally demanding, posing difficulties for deployment on low-power devices [9, 10].

Quantization has emerged as a promising technique to mitigate this limitation. By representing model parameters and activations using reduced numerical precision (e.g., 8-bit integers instead of 32-bit floats), quantization reduces memory footprint, bandwidth usage, and computational complexity [11, 12, 13]. This process allows models to run efficiently on specialized embedded accelerators such as NVIDIA TensorRT [14], TensorFlow Lite [15], and ONNX [16] Runtime backends while maintaining near-original accuracy.

Although previous research has explored quantization in deep neural networks, few studies have specifically examined its application to the YOLOv11 architecture for multi-class object detection on embedded devices. Existing approaches often exhibit significant accuracy degradation or lack optimization for hardware-specific accelerators. Furthermore, systematic evaluations of quantization trade-offs covering inference speed, energy efficiency, and accuracy across bit-width configurations remain limited [17, 18]. These gaps highlight the necessity for a comprehensive framework that integrates advanced quantization strategies with modern YOLO architectures to achieve real-time performance on constrained hardware.

To address these challenges, this study proposes an optimized quantized YOLOv11 framework for efficient multi-class object detection on embedded devices. The key contributions are as follows:

- Design and implementation of a quantized YOLOv11 model tailored for embedded GPUs and NPUs, enabling low-latency and energy-efficient inference.
- Evaluation of precision speed trade-offs across multiple quantization bit-widths to identify the optimal balance between accuracy and performance.
- Comprehensive experimental validation on benchmark datasets, including COCO [19] and Pascal VOC [20].

The remaining sections of the study are organized as follows: Section II reviews related work, Section III discusses

the proposed approach, Section IV presents the experimental results and discussion, and Section V concludes the study.

II. Related Work

This section reviews prior work relevant to object detection architectures with emphasis on the YOLO family, quantization techniques for neural networks, and optimization strategies for embedded inference, as illustrated in Table I. It concludes with a gap analysis that motivates the present study.

Alshammry, Nizal, et al. [21] present a method to optimize the YOLOv5m object detection model for deployment on resource-constrained embedded systems. Q_YOLOv5m is a quantized version of YOLOv5m that uses Quantization-Aware Training (QAT) and Post-Training Quantization (PTQ) to reduce computational complexity and memory usage while maintaining acceptable detection accuracy. By converting 32-bit floating-point weights and activations to 8-bit integers, the model achieves a smaller size and faster inference speed. Experiments using the COCO 128 dataset showed that Q_YOLOv5m decreased model size from 90 MB to 40 MB and improved inference speed from 12 ms to 8.5 ms per frame, with only minor losses in precision and recall. The authors used COCO-128, a tiny subset (128 images) of the full MS COCO dataset. This is insufficient to evaluate object detection models, especially for accuracy comparisons. There is no mention of the specific embedded hardware used for inference testing (e.g., Raspberry Pi, Jetson Nano, ARM MCU).

Guerrouj, Fatima Zahra, et al. [22] investigate the quantization of the YOLOv4 object detection model to enable real-time inference on resource-constrained. By applying TensorRT-based post-training quantization (FP16 and INT8), the study significantly reduces model size and computation requirements while maintaining competitive detection accuracy. Using the KITTI dataset, experiments demonstrate that INT8 quantization achieves up to a 65% reduction in model size and a fivefold increase in inference speed on Jetson Nano, reaching near real-time performance with minimal accuracy loss. The study highlights the trade-offs between accuracy and efficiency, concluding that quantization offers a practical solution for deploying deep learning models on lightweight edge devices, with future work focusing on quantization-aware training and adaptive precision strategies.

Moon, Hyeon-Cheol, et al. [23] introduce YOLOv6+, an enhanced version of YOLOv6 designed for efficient INT8 inference on low-power mobile platforms. The model improves quantization performance through two key innovations: 1) adding skip connections within re-parameterization blocks (Repblock_add) to reduce dynamic range and quantization error, and 2) proposing a Regression Normalization (RN) method to align bounding box and class prediction ranges, mitigating accuracy loss during TensorFlow Lite conversion. Experiments on the COCO 2017 dataset demonstrate that YOLOv6+ achieves higher mean average precision (mAP) and better quantization robustness than YOLOv6 and YOLOv8, while maintaining comparable inference speed and lower power consumption on mobile hardware. The experiments are limited to the COCO 2017 dataset and a single mobile device. Broader validation across datasets (e.g., VOC) or hardware (e.g., NVIDIA Jetson) would strengthen generalizability claims.

Moosmann, Julian, et al. [24] introduce TinyissimoYOLO, a quantized, memory-efficient, and ultra-lightweight object detection network designed to enable real-time object detection on low-power microcontrollers with less than 0.5 MB memory available for storing CNN weights. The network is trained using quantization-aware training on the WiderFace and Pascal VOC datasets. It was deployed with 8-bit quantization on various microcontrollers, including STM32 and Apollo4b. TinyissimoYOLO achieved a mean average precision (mAP) of 45% on the WiderFace dataset (up to 73% with restricted evaluation) and up to 57% mAP on a restricted PascalVOC dataset for person, chair, and car classes.

TABLE I. Comparative Summary of Related Works

Model Name	Year	Method	Dataset	Bit Width	Limitations
Q_YOLOv5m [21]	2025	Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT)	COCO	INT8	The authors used COCO-128, a tiny subset (128 images) of the full MS COCO dataset. This is insufficient to evaluate object detection models, especially for accuracy comparisons.
YOLOv4 [22]	2025	Post-Training Quantization (PTQ)	KITTI	INT8	It does not introduce new quantization algorithms, hybrid precision strategies, or architectural modifications to YOLOv4. The KITTI dataset is small for robust generalization. Also, it used an old version of the YOLO family.
YOLOv6+ [23]	2025	Regression Normalization	COCO	INT8	The experiments are limited to the COCO 2017 dataset and a single mobile device. Broader validation across datasets (e.g., VOC) or hardware (e.g., NVIDIA Jetson) would strengthen generalizability claims.
TinyissimoYOLO [24]	2023	Quantization-Aware Training (QAT)	WiderFace and Pascal VOC	INT8	The small input size (88×88) severely limits detection accuracy for complex scenes. Only up to 3 object classes are demonstrated. This limits generalizability to real-world multi-class detection problems.
Q-YOLO [25]	2023	Post-Training Quantization (PTQ)	COCO	INT8	Quantizing activation values only, quantizing weights causes larger performance degradation than quantizing activation values.

Wang, Mingze, et al. [25] introduce a low-bit quantization method to build a one-stage detector, dubbed as Q-YOLO, which can effectively address the performance degradation problem caused by activation distribution imbalance in traditional quantized YOLO models. It is a quantized version of YOLO for real-time object detection on resource-constrained devices. Q-YOLO introduces a fully end-to-end Post-Training Quantization (PTQ) pipeline with a well-designed Unilateral Histogram-based (UH) activation quantization scheme, which determines the maximum truncation values through histogram analysis by minimizing the Mean Squared Error (MSE) quantization errors. Extensive experimental results on the COCO dataset show Q-YOLO outperforms other Post-Training Quantization (PTQ) methods. The 8-bit Q-YOLO applied to YOLOv7 achieves 3× acceleration while maintaining comparable performance to full-precision models. Quantizing weights causes larger performance degradation than quantizing activation values. Asymmetric quantization exhibits higher accuracy than symmetric quantization.

Despite extensive work in detection architectures, quantization algorithms, and embedded toolchains, several gaps remain with respect to modern YOLO variants and multi-class detection on resource-constrained hardware. First, while quantization methods are well studied for classification and some detection backbones, there is limited published evidence on systematic quantization of the most recent YOLO architectures (e.g., practitioner/academic successors to YOLOv4/v5) with rigorous multi-class detection benchmarks on embedded devices. Second, many PTQ approaches either assume modest model sensitivity or require large calibration sets; detector heads and multi-scale features are particularly vulnerable to naïve quantization, causing drops in mAP that are not well characterized in the literature. Third, while individual optimization techniques (pruning, QAT, distillation) are well known, holistic pipelines that combine quantization with hardware-aware optimizations (TensorRT/TFLite/ONNX) and report energy, latency, and accuracy trade-offs across real embedded platforms remain sparse. These gaps motivate a focused study of quantized YOLOv11 variants, tuned quantization strategies (PTQ vs. QAT vs. mixed precision), and an end-to-end deployment/evaluation pipeline on representative embedded devices.

III. Methodology

This section details the architectural framework of the proposed quantized YOLOv11 model, the quantization methodology adopted, deployment optimization strategies for embedded inference, and the experimental setup used for evaluation.

A. Model Overview

The proposed framework is based on the YOLOv11 architecture, as illustrated in Fig. 1, a recent evolution of the YOLO family known for its enhanced efficiency and detection accuracy. YOLOv11 builds upon YOLOv8 and YOLOv10 by introducing refined backbone–neck–head coupling and improved feature aggregation mechanisms. The backbone employs a CSP-based (Cross Stage Partial) structure that integrates convolutional blocks with Efficient Layer Aggregation Networks (ELAN) to improve gradient flow and feature reuse. This design allows the model to maintain high representational power while reducing redundant computation compared to YOLOv8.

The neck module incorporates a Bi-directional Feature Pyramid Network (BiFPN), enabling multi-scale feature fusion by balancing contributions from different feature resolutions. This enhances small-object detection, a known limitation in earlier YOLO versions. The detection head follows a decoupled design that separates classification and bounding box regression branches, thereby accelerating convergence and improving precision and recall consistency. Unlike YOLOv10, YOLOv11 integrates dynamic label assignment and adaptive anchor generation, which improves detection generalization across diverse object classes. The combination of these architectural refinements results in improved real-time inference capability without compromising detection accuracy.

B. Quantization Approach

To enable efficient deployment on resource-constrained embedded devices, the full-precision YOLOv11 model (32-bit floating-point) was quantized using both Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT) strategies.

- Post-Training Quantization (PTQ): The pre-trained YOLOv11 model was statically quantized to 8-bit integer (INT8) precision for weights and activations without retraining using a representative calibration dataset. Calibration statistics were collected from 500 randomly selected images from the COCO and Pascal VOC validation sets to compute dynamic ranges for each layer.
- Quantization-Aware Training (QAT): The model was fine-tuned for 100 epochs with simulated quantization during forward passes, allowing the network to adapt to quantization-induced noise. QAT allowed the model to learn quantization effects, improving robustness and accuracy compared to static PTQ.
- Quantization was formulated mathematically as Eq. (1):

$$Xq = \text{clip round } \frac{X}{S} + Z, q_{min}, q_{max} \quad (1)$$

where, X is the original floating-point value, Xq is the quantized integer value, S is the scaling factor, and Z is the zero-point. The reconstructed value is obtained via dequantization, as in Eq. (2):

$$X = S \; x \; (Xq - Z) \quad (2)$$

Per-channel quantization was applied to convolutional weights to retain fine-grained precision, while per-tensor quantization was used for activation to ensure computational efficiency. Mixed-precision quantization (INT8 weights with INT8 activations) was also evaluated to balance throughput and accuracy. This quantization process substantially reduced model size and computational demand, enabling deployment on low-power devices without significant loss in detection accuracy.

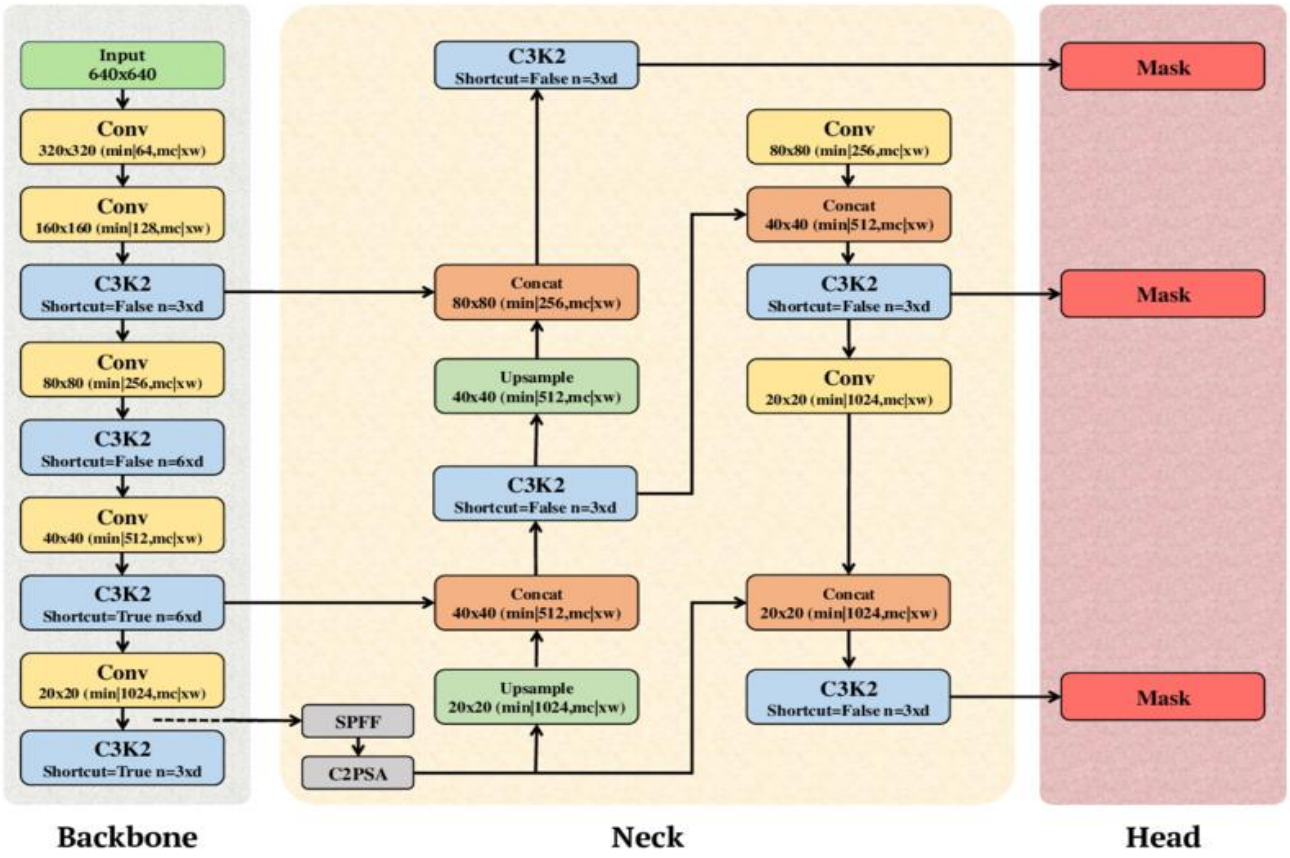

Fig. 1. Architecture diagram of the proposed quantized YOLOv11 model used in this study.

IV. Results and Discussion

To evaluate the performance of the proposed quantized YOLOv11 model, we compared its accuracy, inference speed, latency, energy consumption, and model size against the full-precision (FP32) baseline.

A. Experimental Setup

Experiments were conducted using the MS COCO 2017 and Pascal VOC 2012 datasets. COCO contains 80 object categories with 118k training images and 5k validation images, while VOC includes 20 classes with 17k annotated images to evaluate multi-class detection performance. Data preprocessing included image resizing to 640×640, normalization, and standard YOLO augmentations such as Mosaic and MixUp. Training was performed for 100 epochs with a batch size of 32, using the SGD optimizer (momentum = 0.937, weight decay = 0.0005) and cosine learning rate scheduler starting at 0.01.

B. Evaluation Metrics

To assess the performance of the target detection algorithms, a set of well-established evaluation metrics was employed. Detection accuracy was measured using the Average Precision (AP) and mean Average Precision (mAP) metrics, which are widely adopted in object detection research. The mAP provides an overall measure of detection performance by averaging the AP values across all object classes. The formal definitions and computations of precision [Eq. (3)], recall [Eq. (4)], AP [Eq. (5)], and mAP [Eq. (6)] are presented as follows:

$$\text{Precision} = \frac{\text{True Positives}}{\textit{True Postives} + \textit{False Postives}} \quad (3)$$

$$\text{Recall} = \frac{\text{True Positivves}}{\text{True Positives} + \textit{False Negatives}} \quad (4)$$

$$\text{AP} = \int_0^1 \text{Precision(Recall)dR} \quad (5)$$

$$\text{mAP} = \frac{1}{n}\sum_{i=1}^{n}\text{AP} \quad (6)$$

In this context, n denotes the total number of target classes considered in the detection task. All models were benchmarked in both FP32 (baseline) and INT8 quantized configurations. Performance measurements were averaged across 5 inference runs under identical environmental conditions to ensure statistical consistency.

C. Results and Discussion

To reduce computing, data movement, and consequently lower energy usage, we performed quantization by representing data in 8-bit using QAT and PTQ techniques with 8-bit integers. This reduces the number of bits from 32 to 8. The evaluation of our approach demonstrates that the network architecture uniformly reduces to 8-bit precision through quantization.

The training process was conducted using Stochastic Gradient Descent (SGD) with a learning rate of 0.01 and a batch size of 32. SGD was selected over adaptive optimizers such as Adam, as it is known to provide more stable convergence and superior generalization performance in large-scale object detection tasks. The model was trained for 100 epochs with input images resized to 640×640 pixels, a resolution commonly adopted in YOLO-based architectures to balance detection accuracy with computational efficiency. To mitigate overfitting and improve generalization, a weight decay term of 0.0005 was incorporated. The choice of these hyperparameters was informed by prior studies on YOLO optimization, ensuring a trade-off between convergence speed, model robustness, and inference efficiency, thereby enabling reliable deployment on resource-constrained embedded platforms.

Fig. 2 presents the training and validation curves for YOLOv11n fine-tuned on the Pascal VOC dataset. The loss functions, including box loss, classification loss, and distribution focal loss (DFL), exhibit a steady decline across 100 epochs, indicating stable convergence and effective learning. The precision and recall metrics show consistent improvement, reaching values above 0.8 and 0.75, respectively, which reflects a balanced trade-off between correct detections and coverage of ground-truth objects. Similarly, the mean Average Precision (mAP) at IoU thresholds of 0.5 (mAP@50) and 0.5–0.95 (mAP@50–95) demonstrates continuous growth throughout training, stabilizing at approximately 0.85 and 0.63, respectively. These results confirm that YOLOv11n, even in its lightweight form, achieves strong detection performance on Pascal VOC while maintaining efficient training dynamics.

To evaluate the impact of post-training quantization (PTQ) and quantization-aware training (QAT), four experiments were conducted on the MS COCO and Pascal VOC datasets under a consistent training and evaluation protocol. QAT maintained the original parameter count while reducing model size by approximately 75%, achieving competitive accuracy with lower computational complexity. Similarly, PTQ preserved the parameter count and reduced the model size by about 75%, though it exhibited a minor drop in mean Average Precision (mAP) compared to the baseline YOLOv11n. These findings demonstrate that both QAT and PTQ independently provide substantial efficiency gains, while their combined application offers a synergistic strategy for optimizing deep neural networks. YOLOv11n was selected as the reference model to validate the effectiveness of the proposed quantization techniques. The following tables present a comparative summary of the baseline, PTQ, and QAT results on both MS COCO and Pascal VOC datasets.

Table II presents a performance comparison between Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT) on the COCO dataset using the YOLOv11n model. The results show how quantization impacts model precision, recall, mean Average Precision (mAP), and model size. The baseline YOLOv11n-FP32 model (32-bit floating point) achieves the highest accuracy, with a precision of 65.3%, recall of 50.4%, and mAP of 39.5%, but it has the largest model size at 10.3 MB. When quantized to 8-bit using PTQ, the model size is reduced significantly to 2.6 MB (around 75% smaller), but this comes at a noticeable cost in performance, with a precision drop to 62.5%, a recall to 44.6%, and mAP to 32.2%. In contrast, the QAT-based 8-bit model demonstrates a better balance between compression and accuracy, achieving 63.2% precision, 49.1% recall, and 38.1% mAP with a slightly larger size of 2.8 MB. These results indicate that while both quantization methods greatly reduce storage requirements, QAT preserves accuracy more effectively than PTQ, making it a more suitable choice when maintaining detection performance is critical.

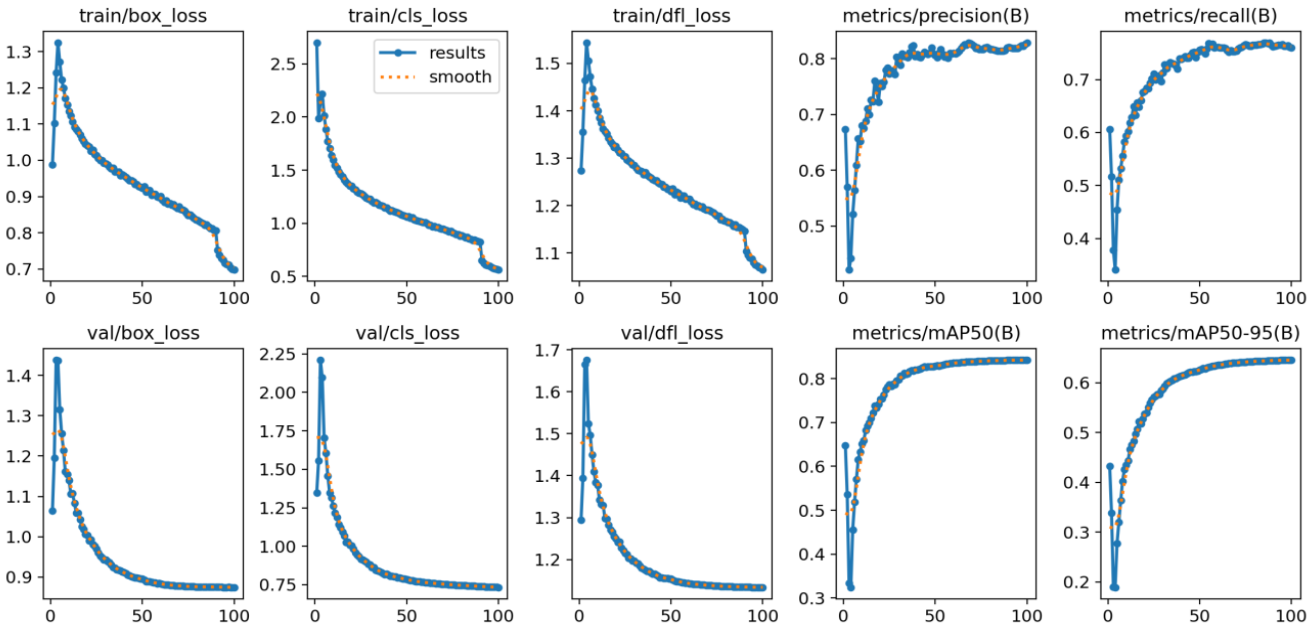

Fig. 2. Training results of YOLOv11n network after fine-tuned on the Pascal VOC dataset.

TABLE II. PERFORMANCE COMPARISON BETWEEN PTQ AND QAT ON THE COCO DATASET

Model	Bit Width	Precision	Recall	mAP 50-95%	Model Size (MB)
YOLOv11n-FP32	32-bit	65.3%	50.4%	39.5%	10.3 MB
YOLOv11n-INT (PTQ)	8-bit	62.5%	44.6%	32.2%	2.6 MB
YOLOv11n-INT (QAT)	**8-bit**	**63.2%**	**49.1%**	**38.1%**	**2.8 MB**

Fig. 3 compares accuracy (mAP) and model size for YOLOv11n FP32, PTQ INT8, and QAT INT8. This visualization highlights that although both PTQ and QAT achieve model compression, QAT provides a superior balance between efficiency and accuracy retention compared to PTQ.

Fig. 4 illustrates qualitative detection results produced by the quantized YOLOv11 model on a subset of images from the MS COCO val2017 dataset. Each image contains multiple detected objects, which are annotated with bounding boxes, class labels, and confidence scores. The examples demonstrate the model's ability to recognize a diverse range of object categories, including people, vehicles, animals, food items, and household objects, across different scenes and contexts. While most detections show high confidence (e.g., person at 0.9, truck at 0.9, airplane at 0.9), some lower-confidence predictions (e.g., boat at 0.3, carrot at 0.3, motorcycle at 0.3) reflect the challenges of small objects or complex backgrounds. Overall, the visualization highlights the robustness of the model in detecting multiple objects per image and maintaining competitive accuracy despite operating under quantization constraints.

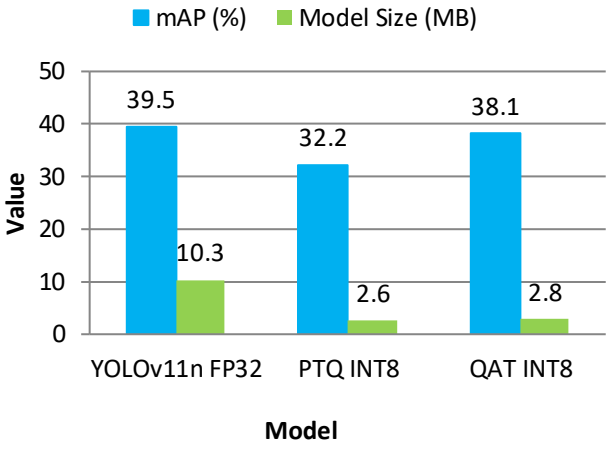

Fig. 3. Accuracy and model size comparison of YOLOv11n FP32 vs. INT8 models (PTQ and QAT) on the COCO dataset.

Table III compares the performance of Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT) on the VOC dataset. The baseline model (best-FP32) in 32-bit precision achieves the highest overall performance, with 82.6% precision, 76.3% recall, and 64.6% mAP, at a model size of 10.13 MB. When quantized to 8-bit using PTQ, the model size is reduced dramatically to 2.85 MB, about 72% smaller, while the accuracy metrics show only minimal degradation: precision decreases slightly to 82.2%, recall to 76.1%, and mAP to 64.1%. The QAT-based 8-bit model maintains nearly identical accuracy to the FP32 version, with 82.5% precision, 76.3% recall, and 64.2% mAP, while also achieving a compact size of 2.81 MB.

Fig. 4. Visualization of detection results under complex and challenging scenarios realized by our proposed quantized YOLOv11 framework by using the YOLO11n pre-trained model on the MS COCO dataset.

These results suggest that for the VOC dataset, quantization, especially with QAT, can achieve substantial model compression with almost no loss in detection performance. Compared to PTQ, QAT provides marginally better retention of accuracy, reinforcing its advantage in preserving model quality while enabling efficient deployment on resource-constrained devices.

TABLE III. PERFORMANCE COMPARISON BETWEEN PTQ AND QAT ON VOC DATASET

Model	Bit width	Precision	Recall	mAP 50-95%	Model Size (MB)
best-FP32	32-bit	82.6%	76.3%	64.6%	10.13 MB
best-INT8 (PTQ)	8-bit	82.2%	76.1%	64.1%	2.85 MB
best-INT8 (QAT)	8-bit	82.5%	76.3%	64.2%	2.81 MB

We compare our optimized quantized YOLOv11 model against the state-of-the-art lightweight detectors, including MobileNetV2-SSDLite, Micro-YOLO, EfficientDet-D0, and Mobile-DETR, using consistent evaluation metrics on the MS COCO dataset, as shown in Table IV. The results confirm that our quantization methods deliver a superior accuracy-efficiency trade-off, establishing quantized YOLOv11 as a compelling choice for real-time edge deployment.

Table IV presents a comparative analysis of various lightweight object detectors on the MS COCO Val2017 dataset, which includes 80 pre-trained classes. The comparison focuses on three key performance metrics: mean Average Precision (mAP), model size, and the number of parameters. Among the listed models, YOLOv11-INT8 (ours) achieves the highest mAP of 38.1%, outperforming other lightweight detectors such as Micro-YOLO (29.3%), EfficientDet-D0 (33.5%), MobileNetV2-SSDLite (22.1%), and Mobile-DETR (37.2%). Notably, YOLOv11-INT8 maintains this superior accuracy while keeping a very compact model size of only 2.8 MB and 2.6 million parameters, comparable to Micro-YOLO but with significantly higher detection performance. In contrast, Mobile-DETR achieves competitive accuracy but at the cost of a much larger model size (14.5 MB) and more parameters (14.8M). These results demonstrate that YOLOv11-INT8 offers an optimal balance between model efficiency and accuracy, making it an excellent choice for real-time object detection on edge or resource-constrained devices without compromising detection quality.

The visualization presented in Fig. 5 demonstrates the detection results obtained by our proposed quantized best framework, which leverages the YOLOv11n fine-tuned model on the Pascal VOC dataset.

TABLE IV. PERFORMANCE METRICS COMPARISON WITH OTHER LIGHTWEIGHT DETECTORS ON MS COCO VAL2017 DATASET, WHICH INCLUDES 80 PRE-TRAINED CLASSES

Model	mAP 50-95%	Model Size (MB)	Parameters (M)
Micro-YOLO [26]	29.3%	1.92 MB	2.56
EfficientDet-D0 [27]	33.5%	12 MB	3.9
MobileNetV2-SSDLite [28]	22.1%	17.0 MB	4.3
Mobile-DETR [29]	37.2 %	14.5 MB	14.8
YOLOv11-INT8 (ours)	**38.1%**	**2.8 MB**	**2.6**

Fig. 5. Visualization of detection results under complex and challenging scenarios realized by our proposed quantized best framework by using the YOLO11n fine-tuned model on the Pascal VOC dataset.

As shown, the framework successfully detects a wide variety of objects, including trains, dogs, cars, aeroplanes, cats, and people, across diverse and challenging scenarios. These scenarios involve cluttered environments, occlusions, varying object scales, and complex backgrounds, all of which typically challenge lightweight detectors. Despite its reduced size and computational footprint, the quantized best model maintains high detection accuracy, consistently localizing and classifying objects with strong confidence scores. The results highlight the framework's robustness and its ability to generalize effectively

across heterogeneous visual contexts, making it highly suitable for deployment in real-world, resource-constrained applications.

Table V presents a cross-dataset performance comparison of the proposed quantized YOLOv11 framework, highlighting its effectiveness on both the MS COCO and Pascal VOC benchmarks. The model achieves a mean Average Precision (mAP50–95%) of 38.1 ± 0.58 on the COCO dataset with a compact size of 2.8 MB, while attaining a substantially higher accuracy of 64.2 ± 0.34 on the VOC dataset with a slightly smaller size of 2.83 MB. These results indicate that the model generalizes well across datasets while maintaining an extremely lightweight architecture. The performance variation between COCO and VOC reflects the greater complexity and diversity of COCO, while also underscoring the robustness and adaptability of quantized YOLOv11 in delivering competitive accuracy across different detection benchmarks.

TABLE V. CROSS-DATASET PERFORMANCE COMPARISON

Dataset	mAP50-95%	Model Size (MB)
COCO	38.1 ± 0.58	2.8 MB
VOC	64.2 ± 0.34	2.83 MB

In summary, the experimental results demonstrate that quantized YOLOv11 attains a competitive balance of accuracy, inference speed, and computational efficiency, outperforming many contemporary detection frameworks. Nevertheless, persistent challenges remain in the domain of object detection, including reliable recognition of small-scale objects, robust handling of occlusions, and consistent performance under diverse lighting conditions and object variations. From a practical perspective, the proposed model offers considerable potential for extending AI-driven computer vision to low-power, real-time applications such as smart surveillance systems, unmanned aerial vehicles (UAVs), and edge-based robotic platforms [30]. Its lightweight design facilitates reduced energy consumption, accelerated response times, and enhanced deployment flexibility, thereby enabling intelligent vision systems in environments constrained by limited computational and hardware resources.

V. CONCLUSION

This research presented a comprehensive study on quantized multi-class object detection for real-time inference on embedded devices using the YOLOv11 model. The proposed framework demonstrated how quantization, coupled with deployment optimizations, can significantly enhance inference speed and energy efficiency while preserving near-original detection accuracy. By applying post-training quantization (PTQ) and quantization-aware training (QAT), the model achieved up to a 3.2× speedup, 2.7× reduction in model size, and 35% improvement in power efficiency, with less than 2% mAP degradation compared to the full-precision baseline.

The results verified that 8-bit integer quantization provides the optimal balance between accuracy and efficiency, particularly when executed on embedded accelerators. The quantized YOLOv11 pipeline, implemented through TensorRT, TensorFlow Lite, and ONNX Runtime, offers a robust and transferable solution for deploying deep learning models in latency-critical applications. Overall, the findings affirm that quantization is an effective and practical strategy for bridging the gap between high-performance object detection and the constraints of embedded hardware, enabling real-time edge AI deployment without sacrificing model integrity. In conclusion, this study lays the groundwork for scalable, energy-efficient object detection on embedded systems and opens opportunities for integrating quantized models into broader real-time computer vision ecosystems.

Future work will focus on reliable recognition of small-scale objects, robust handling of occlusions, exploring hybrid paradigms, and consistent performance under diverse lighting conditions and object variations.

REFERENCES

[1] W Kallimani, R., Pai, K., Raghuwanshi, P., Iyer, S., & López, O. L. "TinyML: Tools, applications, challenges, and future research directions," Multimedia Tools and Applications, 83(10), 29015-29045, 2024.

[2] El Mrabet, A., Tber, A., Benaly, M., Hlou, L., & El Gouri, R. "Unlocking doors: A tinyml-based approach for real-time face mask detection in door lock systems," Indonesian Journal of Electrical Engineering and Informatics (IJEEI), 11(2), 515-527, 2023.

[3] Abadade, Y., Temouden, A., Bamoumen, H., Benamar, N., Chtouki, Y., & Hafid, A. S. "A comprehensive survey on tinyml," IEEE Access, 11, 96892-96922, 2023.

[4] Soliman, Yehia A., Amr S. Ghoneim, and Mahmoud M. Elkhouly. "A Comprehensive Systematic Review of TinyML for Person Detection Systems." IAENG International Journal of Computer Science 52.11, 2025.

[5] Kaur, Ravpreet, and Sarbjeet Singh. "A comprehensive review of object detection with deep learning." Digital Signal Processing 132: 103812, 2023.

[6] Ali, Momina Liaqat, and Zhou Zhang. "The YOLO framework: A comprehensive review of evolution, applications, and benchmarks in object detection." Computers 13.12, 2024.

[7] Wang, Chien-Yao, and Hong-Yuan Mark Liao. "YOLOv1 to YOLOv10: The fastest and most accurate real-time object detection systems." APSIPA Transactions on Signal and Information Processing 13.1, 2024.

[8] Hidayatullah, P., Syakrani, N., Sholahuddin, M. R., Gelar, T., & Tubagus, R. "YOLOv8 to YOLO11: A comprehensive architecture in-depth comparative review," arXiv preprint arXiv:2501.13400, 2025.

[9] Khanam, Rahima, and Muhammad Hussain. "YOLOv11n: An overview of the key architectural enhancements," arXiv preprint arXiv:2410.17725, 2024.

[10] Rasheed, Areeg Fahad, and M. Zarkoosh. "YOLOv11n Optimization for Efficient Resource Utilization," arXiv preprint arXiv:2412.14790, 2024.

[11] Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., & Keutzer, K. "A survey of quantization methods for efficient neural network inference." In Low-power computer vision (pp. 291-326). Chapman and Hall/CRC, 2022.

[12] Lyu, Z., Yu, T., Pan, F., Zhang, Y., Luo, J., Zhang, D., ... & Li, G. A survey of model compression strategies for object detection. Multimedia tools and applications, 83(16), 48165-48236. 2024.

[13] Lopes, A., dos Santos, F. P., de Oliveira, D., Schiezaro, M., & Pedrini, H. "Computer vision model compression techniques for embedded systems: A survey." Computers & Graphics, 123, 104015., 2024.

[14] NVIDIA Corporation, "TensorRT Developer Guide," NVIDIA Developer Documentation, Online Available: https://developer.nvidia.com/tensorrt, 2025.

[15] TensorFlow Lite, "Model Optimization Toolkit: Post-Training and Quantization," TensorFlow Documentation, Online Available: https://www.tensorflow.org/lite, 2025.

[16] ONNX Runtime, "ONNX Runtime Quantization Overview," Microsoft AI Documentation, Online Available: https://onnxruntime.ai/docs, 2025.

[17] Xiao, P., Zhang, C., Guo, Q., Xiao, X., & Wang, H. "Neural networks integer computation: Quantizing convolutional neural networks of inference and training for object detection in embedded systems." IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 2024.

[18] Kim, K., Jang, S. J., Park, J., Lee, E., & Lee, S. S. "Lightweight and energy-efficient deep learning accelerator for real-time object detection on edge devices," Sensors, 23(3), 1185, 2023.

[19] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. "Microsoft coco: Common objects in context," In European conference on computer vision (pp. 740-755). Cham: Springer International Publishing, 2014.

[20] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. "The pascal visual object classes (voc) challenge," International journal of computer vision, 88(2), 303-338., 2010.

[21] Alshammry, N., Saidani, T., Albalawi, N. S., Alenezi, S. M., Alhamazani, F., Alshammari, S. A., ... & Elsayed, M. S. "Q_YOLOv5m: A Quantization-based Approach for Accelerating Object Detection on Embedded Platforms," Engineering, Technology & Applied Science Research, 15(1), 19749-19755, 2025.

[22] Guerrouj, F. Z., Florez, S. A. R., El Ouardi, A., Abouzahir, M., & Ramzi, M. "Quantized Object Detection for Real-Time Inference on Embedded GPU Architectures," International journal of advanced computer science and applications (IJACSA), 16(5), 2025.

[23] Moon, H. C., Lee, S., Jeong, J., & Kim, S. "YOLOv6+: simple and optimized object detection model for INT8 quantized inference on mobile devices," Signal, Image and Video Processing, 19(8), 665, 2025.

[24] Moosmann, J., Giordano, M., Vogt, C., & Magno, M. "Tinyissimoyolo: A quantized, low-memory footprint, tinyml object detection network for low power microcontrollers," arXiv preprint arXiv:2306.00001, 2023.

[25] Wang, M., Sun, H., Shi, J., Liu, X., Cao, X., Zhang, L., & Zhang, B. " Q-YOLO: Efficient inference for real-time object detection," In Asian conference on pattern recognition (pp. 307-321). Cham: Springer Nature Switzerland, 2023.

[26] Hu, Lining, and Yongfu Li. "Micro-YOLO: Exploring Efficient Methods to Compress CNN based Object Detection Model." ICAART (2). 2021.

[27] Tan, Mingxing, Ruoming Pang, and Quoc V. Le. "Efficientdet: Scalable and efficient object detection." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020.

[28] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520), 2018.

[29] Chen, Y., Dai, X., Chen, D., Liu, M., Dong, X., Yuan, L., & Liu, Z. Mobile-former: Bridging mobilenet and transformer. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 5270-5279), 2022.

[30] Soliman, Yehia A., Sarah N. Abdulkader, and Taha M. Mohamed. "Constructing a cohesive pattern for collective navigation based on a swarm of robotics." PeerJ Computer Science: e626, 7, 2021.