A Hardware-Aware Analysis of PTQ and QAT Quantized CNNs for Object Detection on FPGA

Noura Jariri
Laboratory of Electronic Systems Information Processing Mechanics and Energetics, Ibn Tofail University, Kenitra, Morocco
noura.jariri@uit.ac.ma (corresponding author)

Kaoutar Allabouche
Laboratory of Electronic Systems Information Processing Mechanics and Energetics, Ibn Tofail University, Kenitra, Morocco
kaoutar.allabouche@uit.ac.ma

Mohamed Benaly
Innovate Systems Engineering Laboratory (ISI), National School of Applied Sciences of Tetouan (ENSA-Te), Abdelmalek Essaadi University, Tetouan, Morocco
m.benaly@uae.ac.ma

Mohammed Chaman
Laboratory of Electronic Systems Information Processing Mechanics and Energetics, Ibn Tofail University, Kenitra, Morocco
mohammed.chaman@uit.ac.ma

Rania Majdoubi
Laboratory of Electronic Systems Information Processing Mechanics and Energetics, Ibn Tofail University, Kenitra, Morocco
majdoubi.rania@uit.ac.ma

Abdelkader Hadjoudja
Laboratory of Electronic Systems Information Processing Mechanics and Energetics, Ibn Tofail University, Kenitra, Morocco
abdelkader.hadjoudja@uit.ac.ma

Received: 11 January 2026 | Revised: 15 February 2026 | Accepted: 25 February 2026

 DOI: https://doi.org/10.48084/etasr.17475

ABSTRACT

Real-time object detection on embedded platforms is critical for safety-critical and industrial applications, but FPGA deployment remains challenging due to constraints on numerical precision, latency, and hardware resources. Although quantization is widely used to enable efficient FPGA inference, its impact on object-detection models combining classification and bounding-box regression has not been systematically analyzed within an hls4ml based workflow. This work compares Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT) for deploying a lightweight CNN-based detector on an FPGA. An FP32 model is quantized to INT16 and INT8 using QKeras and subsequently converted to fixed-point hardware representations with hls4ml. The results show that PTQ severely degrades detection performance, reducing classification accuracy to approximately 10% and mean IoU below 0.30. In contrast, QAT preserves near-floating-point performance, achieving ≈94% accuracy and ≈0.89 IoU at the software level for both INT16 and INT8. However, default HLS fixed-point configurations introduce software-hardware discrepancies, particularly in classification. A regression-aware refinement that increases fractional precision in the bounding-box head restores hardware-level localization accuracy (IoU ≈0.89), while residual classification gaps remain due to fixed-point constraints. These findings demonstrate

that reliable FPGA-based object detection requires both QAT and hardware-aware fixed-point design, providing practical guidelines for low-precision deployment using hls4ml.

Keywords-quantization; CNN; object detection; PTQ; QAT; hls4ml; FPGA

I. INTRODUCTION

Deep neural networks have become a fundamental component of modern computer vision, enabling robust performance in applications such as autonomous robotics, medical imaging, and edge intelligence[1-3]. Among these tasks, object detection is particularly critical, as it requires not only correct classification but also precise spatial localization under real-time constraints [4]. In safety-sensitive and industrial environments, localization errors or delayed detections can lead to severe consequences, making reliable real-time object detection a key challenge for embedded systems.

Field-Programmable Gate Arrays (FPGAs) have emerged as attractive platforms for deploying deep learning models in such contexts [5, 6] due to their parallel processing capabilities, deterministic latency, and favorable energy efficiency. However, FPGA deployment remains challenging because of limited on-chip memory, constrained computational resources, and the need to replace floating-point arithmetic with fixed-point representations. As a result, deep neural networks must be carefully adapted to operate reliably under reduced numeric precision while preserving detection accuracy[7, 8].

Quantization [9, 10] is one of the most widely used techniques for enabling efficient inference on resource-constrained hardware. By mapping floating-point weights and activations to lower-precision fixed-point or integer representations, quantization significantly reduces memory footprint and computational complexity. Numerous studies have shown that classification networks can maintain high accuracy under INT8 or mixed-precision quantization when appropriate techniques such as calibration or Quantization-Aware Training (QAT) are employed [11, 12]. However, object detection presents additional challenges because it combines classification with bounding-box regression, which requires high numeric sensitivity to preserve geometric consistency [13].

Recent works on quantized object-detection models, including YOLO [14] and SSD [15]-based architectures, report that aggressive Post-Training Quantization (PTQ) often leads to substantial degradation in localization quality, even when classification accuracy remains acceptable [16]. Bounding-box regression outputs are particularly sensitive to quantization noise, rounding, and saturation effects, which can cause unstable or collapsed predictions. These issues are exacerbated in fixed-point hardware implementations, where insufficient fractional precision can further amplify numeric errors.

To mitigate these effects, QAT has been proposed as a more robust alternative to PTQ [17, 18]. By explicitly modeling quantization during training, QAT enables neural networks to adapt their parameters to low-precision arithmetic, thus improving robustness for both classification and regression tasks. Frameworks such as QKeras facilitate this process by supporting hardware-oriented quantization schemes and mixed-precision training. In parallel, High-Level Synthesis (HLS) frameworks have been developed to translate trained neural networks into FPGA-synthesizable implementations. Among them, HLS4ML has gained significant attention for its ability to generate fixed-point, low-latency neural-network implementations directly from high-level models.

Despite numerous studies on FPGA-based classification accelerators and quantized inference using hls4ml [19-21], there is currently no systematic analysis of how PTQ and QAT affect the stability of regression layers in object-detection models within an hls4ml-oriented fixed-point workflow. In particular, the interaction between software-level quantization strategies and hardware-imposed fixed-point precision constraints remains insufficiently understood [13].

The objective of this work is to systematically evaluate quantized CNN-based object detection models deployed on FPGAs using hls4ml. Using the MNIST handwritten digits dataset, FP32, INT16, and INT8 models, trained with PTQ and QAT, this study compares and analyzes the impact of quantization and fixed-point representation on detection performance. The study highlights the limitations of low-precision PTQ and demonstrates the importance of hardware-aware design choices for reliable FPGA-based object detection.

II. METHODOLOGY

As illustrated in Figure 1, the workflow starts with FP32 training using TensorFlow and Keras, followed by PTQ and QAT. The resulting models are converted to fixed-point representations using hls4ml and evaluated at both software and HLS levels to analyze the impact of quantization and fixed-point arithmetic on classification, localization, and software-hardware consistency.

A. Dataset Construction

A synthetic object-detection dataset is derived from the MNIST handwritten digit database. Each digit (28×28) is placed at a random location within a (64×64) image, as shown in Figure 2, introducing an explicit localization task while preserving the original classification difficulty. Digits are inserted on a zero-valued background, with pixel values normalized to the [0, 1] range.

Ground-truth bounding boxes are defined using a normalized center-based representation. Since all digits share identical dimensions, the bounding-box width and height remain constant, while the center position varies. The dataset consists of 60,000 training samples and 10,000 test samples. This controlled setting enables focused analysis of quantization and fixed-point effects on regression stability.

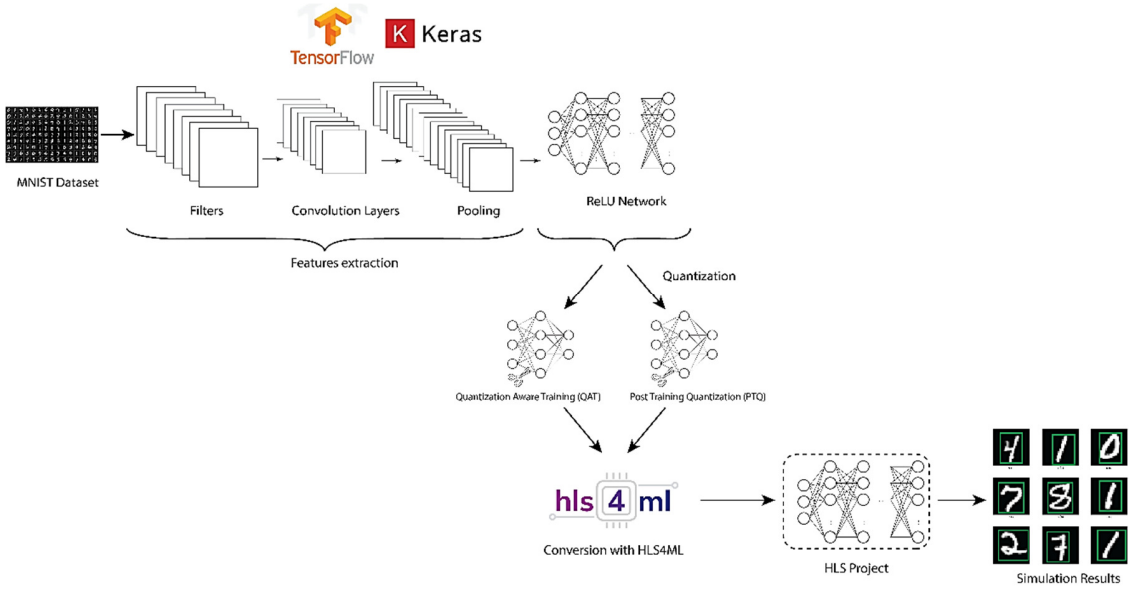

Fig. 1. Overview of the proposed methodology for quantized object detection and HLS deployment.

B. CNN Architecture

The proposed CNN architecture, depicted in Figure 3, is a compact convolutional model designed for hardware efficiency and robustness under quantization. The feature extractor comprises three convolutional blocks with 3×3 convolutions, ReLU activations, and 2×2 max-pooling, with channel counts increasing from 8 to 16 and 32. The resulting feature maps are flattened into a 2048-dimensional vector shared by two task-specific heads. The classification head uses a fully connected layer with 32 units followed by a softmax activation, while the regression head employs a 32-unit fully connected layer with a sigmoid activation to predict the four normalized bounding-box parameters. This dual-head design enables joint optimization of classification and localization within a lightweight detection framework.

C. Training and Quantization Strategies

The network is trained using a multi-task loss combining categorical cross-entropy for classification and Mean Squared Error (MSE) for bounding-box regression, with equal weighting to ensure balanced optimization. Quantization is performed using QKeras, which supports hardware-oriented quantized layers and activations. Four configurations are evaluated: PTQ and QAT at INT16 and INT8 precision. In PTQ, the trained FP32 model is directly quantized without retraining, leading to rounding and clipping effects that can severely impact regression outputs. In contrast, QAT incorporates quantization during training through simulated fixed-point arithmetic and straight-through gradient estimation, allowing the network to adapt its parameters and stabilize low-precision inference.

D. Fixed-Point Representation and Mixed-Precision Design

Beyond software-level quantization, fixed-point arithmetic introduces additional constraints at the hardware level. Although classification is relatively tolerant to reduced precision, bounding-box regression requires fine-grained numerical resolution. Insufficient fractional precision can suppress small spatial variations, resulting in degraded localization. To address this issue, a mixed-precision strategy is applied during hls4ml conversion. Most layers use a uniform fixed-point format optimized for efficiency, while the regression head is assigned increased fractional precision. This selective refinement improves regression stability and software-hardware consistency with minimal impact on overall computational cost.

III. EXPERIMENTAL RESULTS

A. Experimental Setup and Evaluation Protocol

Evaluation was conducted on a fixed subset of 2,000 test samples randomly selected from the MNIST dataset test set and shared across all model configurations, ensuring fair and consistent comparison while keeping HLS simulations tractable. The evaluation covers the complete deployment flow, including an FP32 baseline implemented in Keras, PTQ models at INT16 and INT8 precision, and QAT models at the same precision levels. QAT models are further converted to hardware-oriented fixed-point implementations using hls4ml. Two hardware configurations are considered: the default hls4ml fixed-point assignment and a refined configuration with increased fractional precision in the bounding-box regression head.

Performance is assessed through classification accuracy (Acc), localization quality using mean Intersection-over-Union (IoU) [22, 23], Mean Absolute Error on bounding-box parameters (MAE_{bbox}), Mean Absolute Error on class probabilities (MAE_{cls}), and a software-hardware mismatch rate capturing discrepancies between Keras and HLS predictions.

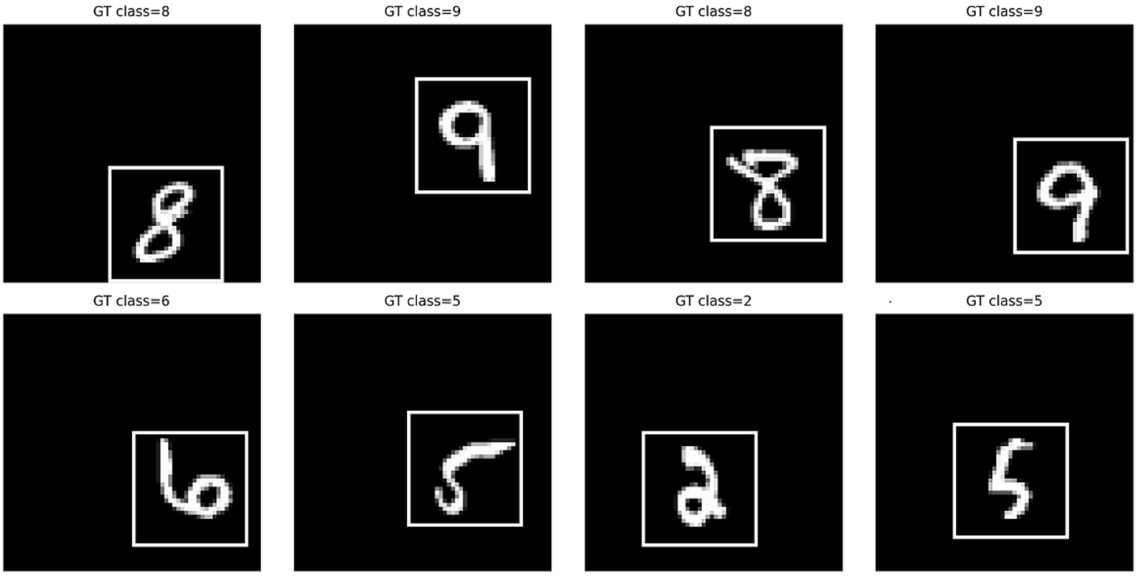

Fig. 2. Samples from the MNIST dataset.

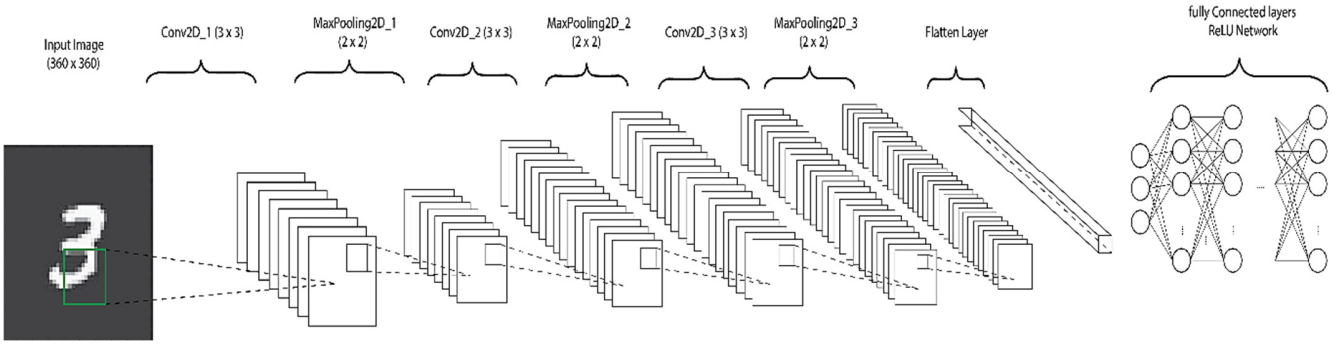

Fig. 3. Architecture of the proposed CNN.

The two essential evaluation metrics are defined as follows:

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FN+FN}} \quad (1)$$

where TP is True Positives, FP is False Positives, TN is True Negatives, and FN is False Negatives.

$$\text{IoU} = \frac{\text{area of overlap}}{\text{area of union}} = \frac{\text{area } (B_P \cap B_{gt})}{\text{area } (B_P \cup B_{gt})} \quad (2)$$

where B_P is the predicted bounding box and B_{gt} is the ground-truth bounding box.

B. Software-Level Quantization Results

Software-level performance was evaluated using Keras inference before hardware mapping. Table I summarizes the quantitative results, with representative qualitative examples shown in Figure 4. The FP32 baseline achieves 93.55% classification accuracy with a mean IoU of 0.92, serving as a reference for both classification and localization. PTQ leads to substantial performance degradation. INT16-PTQ reduces accuracy to 68.4% and IoU to 0.34, while INT8-PTQ further degrades accuracy to 41.2% and IoU to 0.22, accompanied by large increases in MAE_{bbox} and MAE_{cls} indicating unstable localization.

TABLE I. SOFTWARE-LEVEL QUANTIZATION RESULTS (KERAS INFERENCE)

Model	Precision	Accuracy (%)	Mean IoU	MAE_{bbox}	MAE_{cls}
FP32 (baseline)	FP32	93.55	0.920	0.0031	0.0210
PTQ-INT16	16-bit	68.40	0.34	0.0215	0.0840
PTQ-INT8	8-bit	41.20	0.22	0.0338	0.1290
QAT-INT16	16-bit	93.55	0.889	0.0029	0.0260
QAT-INT8	8-bit	93.05	0.890	0.0029	0.0280

In contrast, QAT preserves software-level performance under reduced precision. INT16-QAT matches the FP32 accuracy (93.55%) with a mean IoU of 0.889, while INT8-QAT achieves 93.05% accuracy and a mean IoU of 0.890. For both QAT models, MAE_{bbox} and MAE_{cls} remain low, confirming stable regression and class confidence distributions.

The qualitative examples in Figure 4 corroborate these results: PTQ models exhibit distorted or degenerate bounding boxes, whereas QAT models closely align with ground-truth annotations and the FP32 reference.

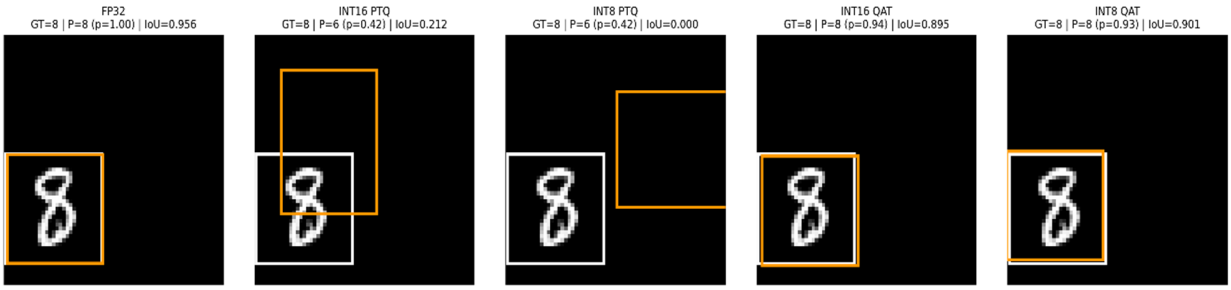

Fig. 4. Impact of quantization at the software level.

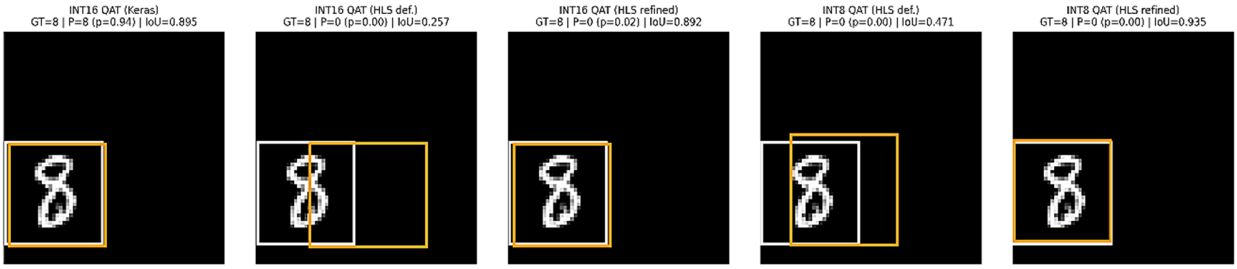

Fig. 5. Effect of fixed-point refinement on HLS inference.

C. Hardware-Level Results

Hardware-level evaluation is performed on QAT models after conversion to fixed-point implementations using hls4ml. All results are obtained via hsl4ml and summarized in Table II, with qualitative comparisons shown in Figure 5.

TABLE II. HARDWARE-LEVEL RESULTS: NON-REFINED VS REFINED FIXED-POINT HLS

Model	Accuracy (%)	IoU	MAE_{cls}	MAE_{bbox}	Mismatch (%)
INT16 QAT (HLS default)	69.40	0.249	0.0788	0.1503	28.90
INT16 QAT (HLS refined)	74.50	0.886	0.0688	0.0085	23.70
INT8 QAT (HLS default)	74.95	0.275	0.0680	0.1553	23.00
INT8 QAT (HLS refined)	73.25	0.889	0.0712	0.0088	24.60

Using the default hls4ml fixed-point configuration, significant degradation is observed. INT16-QAT accuracy drops to 69.40% with a mean IoU of 0.249 and a mismatch rate of 28.9%, indicating strong divergence between Keras and HLS predictions. Similar behavior is observed for INT8-QAT, with 74.95% accuracy, IoU of 0.275, and a 23.0% mismatch rate. Applying the proposed bbox-aware refined fixed-point configuration substantially restores localization performance. For INT16-QAT, mean IoU increases from 0.249 to 0.886 and MAE_{bbox} decreases from 0.1503 to 0.0085. INT8-QAT exhibits similar improvements, with IoU increasing from 0.275 to 0.889 and MAE_{bbox} reduced to 0.0088.

Despite the recovery of localization accuracy, classification accuracy remains limited at the hardware level, reaching 74.50% for INT16-QAT and 73.25% for INT8-QAT, with mismatch rates above 23%. This indicates that fixed-point effects continue to impact class decision consistency even after regression-oriented precision refinement.

The qualitative results in Figure 5 confirm these trends: default configurations produce visibly inaccurate bounding boxes, while the refined configuration yields geometrically consistent localization, although residual class mismatches remain observable.

IV. DISCUSSION

The experimental results reveal a fundamentally different sensitivity of object-detection models to quantization compared to pure classification networks. While classification decisions can tolerate moderate numerical perturbations, bounding-box regression relies on the accurate representation of continuous spatial parameters. This distinction explains the contrasting behaviors observed under PTQ, QAT, and hardware-oriented fixed-point inference. At the software level, the results confirm that PTQ is inadequate for detection models involving regression outputs. Without retraining, PTQ introduces rounding and clipping effects that distort activation distributions, particularly in the regression branch, leading to unstable or degenerate bounding-box predictions even at INT16 precision. The resulting degradation in IoU and regression error demonstrates that geometric consistency cannot be preserved through PTQ alone. QAT significantly mitigates these effects by explicitly modeling quantization during optimization. By simulating fixed-point arithmetic in the forward pass and allowing parameters to adapt through gradient-based learning, QAT aligns weight distributions and activation ranges with low-precision representations. Consequently, INT16-QAT and INT8-QAT models achieve software-level performance close to the FP32 baseline, maintaining high classification accuracy and stable localization.

However, hardware-level evaluation shows that QAT alone is insufficient when models are mapped to fixed-point implementations in FPGA-oriented workflows. Default HLS fixed-point configurations impose additional constraints on

numerical range and fractional resolution that are not optimized during software training. In particular, insufficient fractional precision in regression layers limits the representation of small spatial variations, resulting in significant divergence between Keras and HLS predictions despite identical quantized weights.

The proposed bbox-aware fixed-point refinement addresses this limitation by selectively increasing the fractional precision of the regression head while keeping the remainder of the network unchanged. This targeted adjustment restores numerical resolution where it is most critical, substantially reducing software-hardware mismatch and preserving localization accuracy at the HLS level. These results indicate that reliable FPGA-oriented object detection requires a joint consideration of quantization strategy and hardware-level numeric design.

Compared to existing works, this study presents a unified evaluation that jointly examines PTQ, QAT, and HLS-level fixed-point effects in the context of object detection. Most FPGA-based detection pipelines rely on custom HDL implementations or specialized optimization frameworks and primarily emphasize throughput and energy efficiency, while lacking a systematic analysis of quantization-induced instability in bounding-box regression [24, 25]. Other studies explore fixed-point or mixed-precision strategies on FPGAs but focus mainly on classification tasks or generic CNN acceleration, without addressing quantized object detection or software-hardware consistency [26-29]. In contrast, this work demonstrates that hardware-level localization fidelity is jointly governed by the chosen quantization strategy and a regression-aware fixed-point design within an hls4ml-based workflow.

TABLE III. COMPARISON WITH RELATED FPGA-BASED QUANTIZATION STUDIES

Ref.	Task	Quantization	FPGA flow	Regression	Main limitation
[24]	Classification	PTQ (INT8)	TFLite + FPGA	No	No regression, no QAT
[25]	YOLO detection	ADMM	Custom HDL	Yes	No PTQ/QAT comparison
[26]	Classification	Mixed	Custom FPGA	No	No detection task
[27]	Classification	QAT (variable bits)	FPGA-centric	No	No localization analysis
[28]	Classification	QNN (INT)	Custom accelerator	No	No detection, no IoU
[29]	Classification	Quantized CNN	FINN	No	No bounding-box regression
This work	Detection	PTQ+ QAT	HLS4ML	Yes	Unified PTQ/QAT + HLS drift analysis

Despite these promising results, this study is limited to a simplified detection task that involves single objects with fixed bounding-box dimensions. While this controlled setting enables clear isolation of quantization and fixed-point effects, more complex scenarios with multiple objects and varying scales may introduce additional challenges. Moreover, the evaluation is restricted to HLS C-level simulation and does not include full FPGA synthesis or hardware performance metrics, which will be addressed in future developments.

V. CONCLUSION

This work analyzed the impact of quantization strategies and fixed-point arithmetic on a lightweight CNN for object detection within an HLS-oriented workflow. The results demonstrate that PTQ is not suitable for detection models involving bounding-box regression, as it leads to severe localization instability under reduced numerical precision. In contrast, QAT preserves both classification accuracy and localization performance.

Beyond training-level quantization, this study shows that hardware-oriented fixed-point design plays a critical role in FPGA deployment. Default HLS precision settings introduce significant software-hardware mismatches, particularly in regression layers. By selectively increasing the fractional precision of the bounding-box head, consistent behavior between Keras and HLS inference is restored without increasing precision across the entire network.

Overall, these findings highlight that reliable FPGA-based object detection requires a joint consideration of QAT and hardware-aware fixed-point design.

REFERENCES

[1] S. Pouyanfar *et al.*, "A Survey on Deep Learning: Algorithms, Techniques, and Applications," *ACM Computing Surveys*, vol. 51, no. 5, pp. 1–36, Sept. 2019, https://doi.org/10.1145/3234150.

[2] S. Dargan, M. Kumar, M. R. Ayyagari, and G. Kumar, "A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning," *Archives of Computational Methods in Engineering*, vol. 27, no. 4, pp. 1071–1092, Sept. 2020, https://doi.org/10.1007/s11831-019-09344-w.

[3] S. Dong, P. Wang, and K. Abbas, "A survey on deep learning and its applications," *Computer Science Review*, vol. 40, May 2021, Art. no. 100379, https://doi.org/10.1016/j.cosrev.2021.100379.

[4] J. He, J. Jiang, and C. Zhang, "A survey of lightweight methods for object detection networks," *Array*, vol. 29, Mar. 2026, Art. no. 100589, https://doi.org/10.1016/j.array.2025.100589.

[5] S. H. Hozhabr and R. Giorgi, "A Survey on Real-Time Object Detection on FPGAs," *IEEE Access*, vol. 13, pp. 38195–38238, 2025, https://doi.org/10.1109/ACCESS.2025.3544515.

[6] T. Saidani, R. Ghodhbani, A. Alhomoud, A. Alshammari, H. Zayani, and M. Ben Ammar, "Hardware Acceleration for Object Detection using YOLOv5 Deep Learning Algorithm on Xilinx Zynq FPGA Platform," *Engineering, Technology & Applied Science Research*, vol. 14, no. 1, pp. 13066–13071, Feb. 2024, https://doi.org/10.48084/etasr.6761.

[7] B. Jacob *et al.*, "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference." arXiv, 2017, https://doi.org/10.48550/ARXIV.1712.05877.

[8] G. Alsuhli, V. Sakellariou, H. Saleh, M. Al-Qutayri, B. Mohammad, and T. Stouraitis, "A Survey and Comparative Analysis of Number Systems for Deep Neural Networks," *Proceedings of the IEEE*, vol. 113, no. 2, pp. 172–207, Feb. 2025, https://doi.org/10.1109/JPROC.2025.3578756.

[9] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A Survey of Quantization Methods for Efficient Neural Network Inference." arXiv, 2021, https://doi.org/10.48550/ARXIV.2103.13630.

[10] L. Wei, Z. Ma, C. Yang, and Q. Yao, "Advances in the Neural Network Quantization: A Comprehensive Review," *Applied Sciences*, vol. 14, no. 17, Aug. 2024, Art. no. 7445, https://doi.org/10.3390/app14177445.

[11] H. C. Moon, S. Lee, J. Jeong, and S. Kim, "YOLOv6+: simple and optimized object detection model for INT8 quantized inference on mobile devices," *Signal, Image and Video Processing*, vol. 19, no. 8, Aug. 2025, Art. no. 665, https://doi.org/10.1007/s11760-025-04234-0.

[12] D. Wu, Y. Wang, Y. Fei, and G. Gao, "A Novel Mixed-Precision Quantization Approach for CNNs," *IEEE Access*, vol. 13, pp. 49309–49319, 2025, https://doi.org/10.1109/ACCESS.2025.3551802.

[13] L. Huang *et al.*, "HQOD: Harmonious Quantization for Object Detection." arXiv, 2024, https://doi.org/10.48550/ARXIV.2408.02561.

[14] A. Chen, "Comparative Analysis of YOLO Variants Based on Performance Evaluation for Object Detection," *ITM Web of Conferences*, vol. 70, 2025, Art. no. 03008, https://doi.org/10.1051/itmconf/20257003008.

[15] A. Kumar and S. Srivastava, "Object Detection System Based on Convolution Neural Networks Using Single Shot Multi-Box Detector," *Procedia Computer Science*, vol. 171, pp. 2610–2617, 2020, https://doi.org/10.1016/j.procs.2020.04.283.

[16] M. Wang, H. Sun, J. Shi, X. Liu, B. Zhang, and X. Cao, "Q-YOLO: Efficient Inference for Real-time Object Detection." arXiv, 2023, https://doi.org/10.48550/ARXIV.2307.04816.

[17] Z. Jiang, C. Li, T. Qu, C. He, and D. Wang, "MSQuant: Efficient Post-Training Quantization for Object Detection via Migration Scale Search," *Electronics*, vol. 14, no. 3, Jan. 2025, Art. no. 504, https://doi.org/10.3390/electronics14030504.

[18] C. U. Oflamaz and M. E. Yalçın, "HADQ-Net: A Power-Efficient and Hardware-Adaptive Deep Convolutional Neural Network Translator Based on Quantization-Aware Training for Hardware Accelerators," *Electronics*, vol. 14, no. 18, Sept. 2025, Art. no. 3686, https://doi.org/10.3390/electronics14183686.

[19] T. Aarrestad *et al.*, "Fast convolutional neural networks on FPGAs with hls4ml," *Machine Learning: Science and Technology*, vol. 2, no. 4, Dec. 2021, Art. no. 045015, https://doi.org/10.1088/2632-2153/ac0ea1.

[20] S. Curzel, N. Ghielmetti, M. Fiorito, and F. Ferrandi, "De-specializing an HLS library for Deep Neural Networks: improvements upon hls4ml." arXiv, Mar. 24, 2021, https://doi.org/10.48550/arXiv.2103.13060.

[21] F. Fahim *et al.*, "hls4ml: An Open-Source Codesign Workflow to Empower Scientific Low-Power Machine Learning Devices." arXiv, Mar. 23, 2021, https://doi.org/10.48550/arXiv.2103.05579.

[22] H. Zhu, H. Wei, B. Li, X. Yuan, and N. Kehtarnavaz, "A Review of Video Object Detection: Datasets, Metrics and Methods," *Applied Sciences*, vol. 10, no. 21, Nov. 2020, Art. no. 7834, https://doi.org/10.3390/app10217834.

[23] R. Padilla, S. L. Netto, and E. A. B. Da Silva, "A Survey on Performance Metrics for Object-Detection Algorithms," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, July 2020, pp. 237–242, https://doi.org/10.1109/IWSSIP48289.2020.9145130.

[24] O. B. H. Salah, S. Messaoud, M. A. Hajjaji, M. Atri, and N. Liouane, "Post-training quantization for efficient FPGA-based neural network acceleration," *Integration*, vol. 105, Nov. 2025, Art. no. 102508, https://doi.org/10.1016/j.vlsi.2025.102508.

[25] C. Ding, S. Wang, N. Liu, K. Xu, Y. Wang, and Y. Liang, "REQ-YOLO: A Resource-Aware, Efficient Quantization Framework for Object Detection on FPGAs." arXiv, 2019, https://doi.org/10.48550/ARXIV.1909.13396.

[26] S. E. Chang *et al.*, "Mix and Match: A Novel FPGA-Centric Deep Neural Network Quantization Framework." arXiv, 2020, https://doi.org/10.48550/ARXIV.2012.04240.

[27] C. Sun *et al.*, "HGQ: High Granularity Quantization for Real-time Neural Networks on FPGAs," in *Proceedings of the 2026 ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Feb. 2026, pp. 79–91, https://doi.org/10.1145/3748173.3779200.

[28] M. Tasci, A. Istanbullu, V. Tumen, and S. Kosunalp, "FPGA-QNN: Quantized Neural Network Hardware Acceleration on FPGAs," *Applied Sciences*, vol. 15, no. 2, Jan. 2025, Art. no. 688, https://doi.org/10.3390/app15020688.

[29] M. Jaiswal, V. Sharma, A. Sharma, S. Saini, and R. Tomar, "Quantized CNN-based efficient hardware architecture for real-time hand gesture recognition," *Microelectronics Journal*, vol. 151, Sept. 2024, Art. no. 106345, https://doi.org/10.1016/j.mejo.2024.106345.