Indonesian Journal of Electrical Engineering and Computer Science
Vol. 40, No. 3, December 2025, pp. 1567~1575
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v40.i3.pp1567-1575

Optimizing YOLOv8: OpenVINO standard quantization vs accuracy-controlled for edge deployment

Chandrakala G. Raju[1], Ajaykumar Devarapalli[2], Rakshitha Mahendran[1], Sathwik Madhusudan[1], Omkar Prasad[1]
[1]Department of Information Science and Engineering, BMS College of Engineering (BMSCE), Bangalore, India
[2]Department of Electronics and Instrumentation Engineering, BMS College of Engineering (BMSCE), Bangalore, India

Article Info

Article history:

Received Mar 4, 2025
Revised Oct 24, 2025
Accepted Nov 16, 2025

Keywords:

Accuracy control
OneAPI
OpenVINO
Quantization
Smart queue management
YOLOv8

ABSTRACT

Object detection models, such as you only look once (YOLO), are widely utilized for real-time applications; however, their computational complexity often restricts deployment on edge devices. This research investigates the optimization of YOLO models using OpenVINO, both with and without accuracy control, to enable efficient inference while preserving model accuracy. A two-step pipeline is proposed: first, YOLO models are converted into OpenVINO's intermediate representation (IR) format, followed by the application of post-training quantization (PTQ) to reduce model size and enhance latency. Additionally, an accuracy-aware quantization approach is introduced, which maintains model performance by calibrating with a validation dataset. Experimental results illustrate the trade-offs between standard and accuracy-controlled quantization, demonstrating improvements in inference speed while ensuring minimal accuracy degradation. This study provides a practical framework for deploying lightweight object detection models on edge devices, particularly in real-world scenarios such as autonomous systems, smart surveillance, and smart queue management systems.

Corresponding Author:

Ajaykumar Devarapalli
Department of Electronics and Instrumentation Engineering, BMS College of Engineering (BMSCE)
P.O. Box No 1908, Bull Temple Road, Basavanagudi, Bangalore – 560019, Karnataka, India
Email: d.ajay402@gmail.com

1. INTRODUCTION

In the era of high-performance computing and artificial intelligence (AI), Intel's oneAPI [1] has emerged as a unified, open-standard programming model designed to maximize hardware efficiency across central processing units (CPUs), graphics processing units (GPUs), field-programmable gate arrays (FPGAs), and other accelerators. Unlike solutions that limit flexibility, oneAPI enables developers to write optimized, cross-architecture code without rewriting algorithms. It serves industries such as autonomous driving, healthcare, and financial services powering workloads from data processing to deep learning (DL).

One key component of the oneAPI toolkit is open visual inference and neural network optimization (OpenVINO™), designed to accelerate DL models for edge devices and heterogeneous computing environments [2]. OpenVINO optimizes inference tasks by converting models from frameworks like TensorFlow, PyTorch, and ONNX into an optimized format that runs efficiently on various Intel hardware.

DL models often face challenges due to large size and high computational cost. Full-precision (FP32) models consume heavy memory and power, making them unsuitable for real-time use. Quantization addresses this by reducing numerical precision (e.g., FP32 → INT8), boosting speed and lowering power

consumption [3]. However, traditional quantization methods often struggle to retain accuracy, motivating research into advanced optimization methods.

Han *et al.* [4] introduced deep compression (pruning, quantization, and Huffman coding) to reduce the memory footprint of neural networks greatly reducing size but requiring retraining which limits its real-time deployment. Choi *et al.* [5] proposed reinforcement learning-based quantization this approach is computationally expensive, requiring a complex training pipeline. Rastegari *et al.* [6] proposed XNOR-Net, using bitwise operations for speed but losing accuracy on complex datasets. Kulkarni *et al.* [7] surveyed quantization strategies, identifying post-training quantization (PTQ) as practical though prone to accuracy loss.

Several researchers have explored hardware-aware optimizations for quantization. Mishra *et al.* [8] proposed wide reduced-precision networks (WRPN), which increase the number of channels while using low-bit precision. Wu *et al.* [9] analyzed integer quantization techniques, comparing their efficiency across different hardware platforms. They concluded that integer quantization provides significant speedup with minimal accuracy loss, though certain activation functions remain difficult to quantize. Future work includes designing activation functions that are more quantization-friendly. He *et al.* [10] introduced an AutoML-based quantization approach to optimize DL models on mobile devices, showing notable improvements in performance. However, their approach required a significant amount of computational resources during optimization, indicating a need for more efficient AutoML-based quantization methods.

In object detection, Chen *et al.* [11] improved you only look once (YOLOv5) using pruning and quantization, achieving faster inference but with hardware-dependent gains. Chebtha *et al.* [12] used OpenVINO-based quantization for drone detection, showing speed benefits yet limited testing. Future research should focus on hybrid and adaptive precision methods to balance speed and accuracy.

Lastly, quantization techniques have been integrated into industry applications such as smart queue management and edge computing. Demidovskij *et al.* [13] presented OpenVINO DL Workbench, which optimizes neural network inference through model compression and quantization. The tool significantly improves execution speed, yet its reliance on Intel hardware limits its general applicability. Dagli and Eken [14] deployed a smart queuing system on edge devices using OpenVINO, demonstrating efficient inference capabilities for real-time applications. However, their study primarily focused on Intel-based edge hardware, necessitating further evaluations on heterogeneous computing environments. Schaefer *et al.* [15] explored mixed-precision quantization to accelerate edge AI applications, showing that dynamic precision selection enhances efficiency while minimizing accuracy degradation. However, further studies are required to standardize quantization strategies for diverse AI workloads. Kummer *et al.* [16] introduced adaptive precision control in quantization-aware training, mitigating accuracy loss without retraining. Future work involves extending these techniques to support a wider range of neural architectures.

2. METHOD

The quantization process is built on OpenVINO, a toolkit that enables efficient deployment of DL models across diverse hardware platforms. In this research, the YOLOv8 model [17] undergoes two key stages: (i) conversion from its original PyTorch format to OpenVINO's intermediate representation (IR) for compatibility and optimized execution on Intel hardware, and (ii) optimization through quantization. The neural network compression framework (NNCF), part of OpenVINO, provides multiple quantization techniques such as large language model (LLM) weight compression, training-time optimization, and PTQ. While weight compression benefits LLMs, it is less suitable for object detection, and training-time methods like quantization aware training and pruning require retraining in the original framework. Figure 1 illustrates the fundamental impact of quantization on a deep neural network [18].

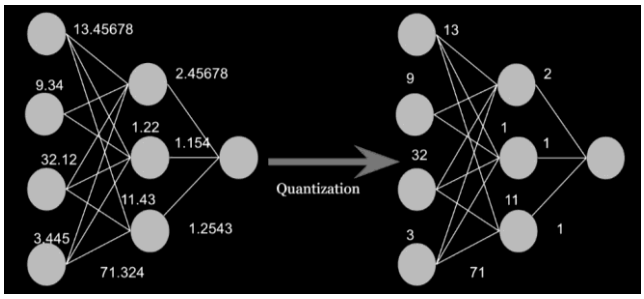

Figure 1. Quantization process

PTQ [19] converts a high-precision (FP32) model into a lower precision (INT8) format, reducing memory usage and accelerating inference without any retraining. Using the COCO dataset as a calibration set, the NNCF framework analyzes activation ranges to determine optimal scaling factors (FP32 → INT8), aided by preprocessing transformations on the dataset to meet the quantization requirements. A mixed-precision [20] approach ensures performance critical layers remain in higher precision, and the resulting quantized model is saved for inference and benchmarking, as illustrated in Figure 2.

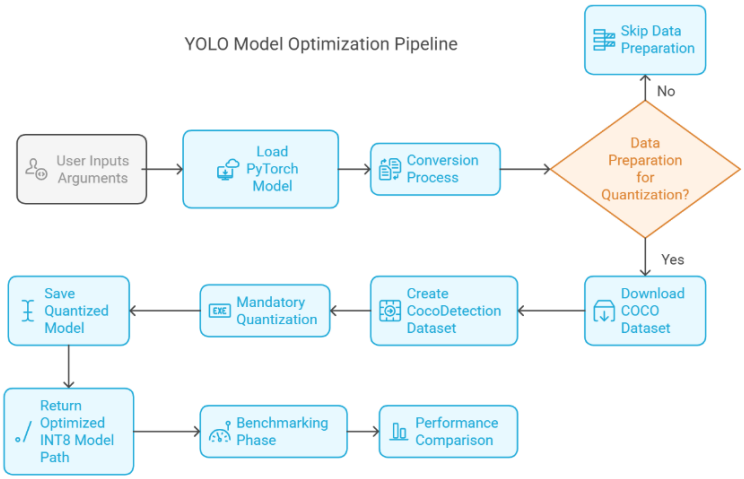

Figure 2. YOLO quantization

Quantization can lead to a drop in accuracy [21], which is why the accuracy control mechanism is added to the NNCF's quantization. The PTQ (INT8) with accuracy control is used to achieve an optimal balance between performance and precision. The COCO dataset is used again to extract and format the images and labels before it is split into two subsets: the calibration dataset, used for model quantization, and the validation dataset, used to evaluate model accuracy. Figure 3 shows a high level view of quantization with accuracy control.

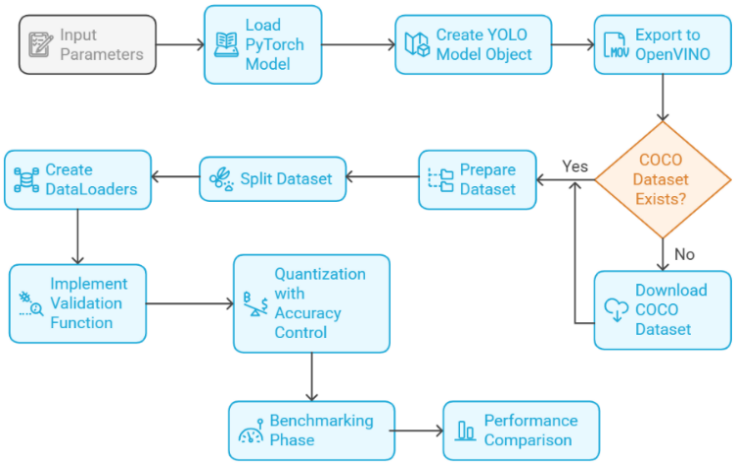

Figure 3. YOLO quantization with accuracy control

To ensure accuracy is maintained after optimization, model validation is performed by running batches of images through the compiled OpenVINO model after every iteration. Detection of confidence scores, specifically for the "person" class, are extracted and compared against ground-truth annotations from COCO. The performance of the model is then assessed using the mean average precision (mAP) metric.
Confidence score calculation:

$$\text{person\_scores} = \text{conf\_scores} \times (\text{class\_scores} > 0.5) \quad (1)$$

- conf_scores: the raw confidence score for each detected object.
- class_scores: the probability that the detected object belongs to the "person" category.
- (class_scores >0.5): a Boolean mask (1 if the object is a person, 0 otherwise)

Before quantization, a baseline measurement is established by evaluating the original floating point 32 bit YOLO model on the validation dataset, computing its mAP score. This serves as a reference to compare accuracy after quantization. Since accuracy control is enabled during quantization, the model undergoes an iterative process where the mAP score is recalculated after each iteration. If the accuracy drop exceeds the acceptable threshold of 1% which has been manually set using the max_drop hyperparameter (0.01); adjustments, such as mixed-precision quantization are applied to maintain accuracy.

In (2) shows how the validation function will be calculating the mAP from the validation dataset after each iteration.

- mAP calculation

mAP is computed using the precision-recall curve:
Precision (P)

$$P = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)+False Positives (FP)}} \quad (2)$$

Measures how many detected "person" objects are actually correct

After quantization, the final model is tested on the validation dataset, and its mAP score is compared against the original FP32 model. The quantization process stops when the accuracy drop remains within the 1% threshold, ensuring minimal accuracy loss.

- Average precision (AP)

AP measures the area under the precision-recall curve for a specific class (in this case, the "person" category). It provides a single metric summarizing the precision-recall tradeoff.

$$AP = \int_0^1 P(R)\, dR \quad (3)$$

Where,
P(R): precision as a function of recall. It shows how precision changes with recall.
R: recall, ranging from 0 to 1.
$\int_0^1$: the integral from 0 to 1 calculates the total area under the precision-recall curve, representing the overall performance for the class.

- mAP

mAP computes the mean of the AP scores across all object classes in the dataset.

$$mAP = \frac{1}{C}\sum_{c=1}^{C} AP_c \quad (4)$$

Since this function only deals with person detection, $C = 1$, making mAP equivalent to AP
Where:
C: the total number of object classes.
AP_c: the average precision for class c.
$\sum_{c=1}^{C}$: summation of the AP scores for all classes from $c = 1$ to $c = C$.
$\frac{1}{C}$: averages the AP scores by dividing the total by the number of classes.

To assess performance improvements, the models undergo benchmarking using OpenVINO's benchmarking tools [22]. The benchmarking process systematically measures performance metrics, such as latency, throughput, and inference speed across different hardware devices, such as CPUs and GPUs, to optimize real-world applications. Three versions of the model are evaluated: the full-precision FP32 model, the INT8 quantized model, and an INT8 model with accuracy control. This allows for assessing the trade-off between speed and accuracy of various models.

The benchmarking process evaluates the model's inference performance on randomly generated input data batches for 60 seconds using a CPU. As illustrated in Figure 4, the process follows 11 essential steps to ensure accurate performance measurement. The process begins by parsing user-defined parameters (model path, device type, precision, and batch size) and initializing OpenVINO's runtime to configure the chosen inference device with optimized settings, load the model, validate inputs, and compile it for

execution. Performance is then measured through multiple inference iterations using parallel requests evaluating key metrics such as latency (average time per inference) and throughput (frames processed per second).

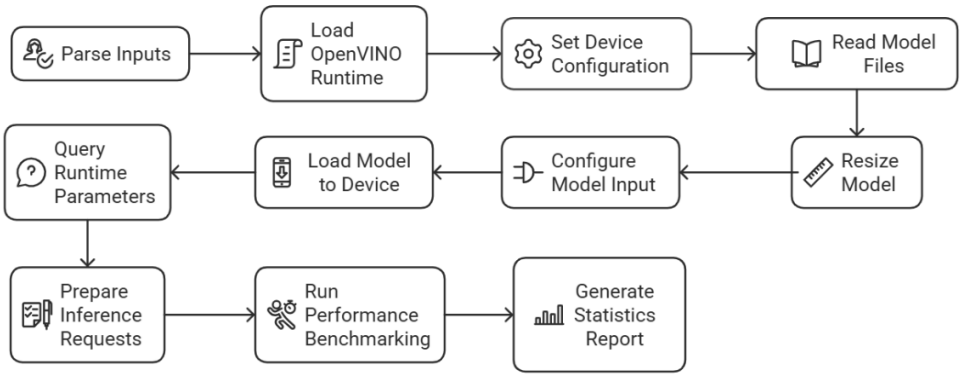

Figure 4. Benchmarking steps

These values are computed using the following equations:

$$L = \frac{D}{N} \tag{5}$$

where,
L: is the average inference latency per frame
D: is the total inference duration (ms)
N: is the total number of inferences completed

For batch processing, the batch latency is computed as:

$$L_B = \frac{L}{B} \tag{6}$$

where B is the batch size.

$$T = \frac{N}{D} \times 1000 \tag{7}$$

Where T represents throughput in FPS

This methodology ensures an accurate evaluation of the model's efficiency across various execution conditions, with results such as inference count, duration, and latency compiled into performance reports for hardware comparison. A trade-off between latency and throughput guides performance tuning, reducing parallel streams minimizes latency while increasing them maximizes throughput. The optimized model is then integrated into an intelligent queue management system [23] for real-time crowd monitoring and resource optimization [24], achieving a balance between speed, memory efficiency, and accuracy suitable for real-world deployment [25]. Using OpenVINO runtime with AUTO device selection, the model automatically utilizes the best available hardware for optimal performance, as illustrated in Figure 5. Incoming video frames are preprocessed to meet YOLOv8 input requirements through letterbox resizing (maintaining aspect ratio with padding), normalization (scaling pixel values to 0–1), format conversion (HWC to CHW), and batch dimension expansion to align with model specifications.

Each preprocessed frame is fed into the OpenVINO optimized YOLOv8 model for real-time object detection, where raw predictions are refined using non-maximum suppression (NMS), box rescaling, and class filtering to extract people-related detections for queue analysis. These detections are mapped onto predefined zones from a JSON configuration, enabling the system to count individuals per zone, track crowd fluctuations over time, and trigger alerts when queue thresholds are exceeded. Annotated frames displaying bounding boxes, queue statistics, alerts, and performance metrics are rendered in real time, ensuring efficient, optimized queue management that enhances customer experience and operational efficiency.

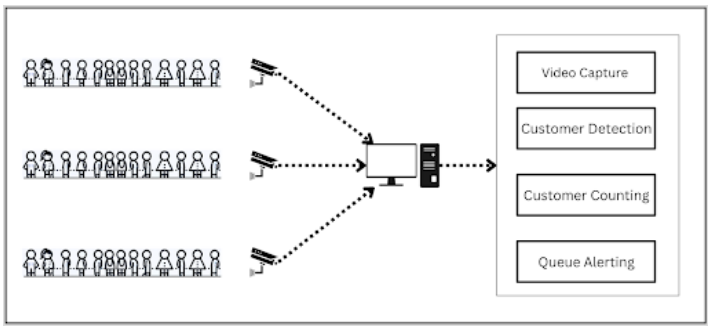

Figure 5. Queue management system

3. RESULTS AND DISCUSSION

The benchmarking results provide an analysis of the impact of model precision, device type, and dataset split on inference performance. Key observations from Table 1 are:

Table 1. Benchmarking for CPU

Dataset split	Precision	Count (CPU/GPU)	Duration (ms) (CPU/GPU)	Median latency (ms) (CPU/GPU)	Average latency (ms) (CPU/GPU)	Throughput (FPS) (CPU/GPU)
70-30	FP32	89 / 821	30370.29 / 30048.85	389.25 / 36.41	341.14 / 36.47	2.93 / 27.32
70-30	INT8	323 / 1416	30144.20 / 30038.45	99.24 / 20.96	93.23 / 21.08	10.72 / 47.14
70-30	INT8_AC	293 / 1456	30098.76 / 30018.94	99.12 / 20.45	102.63 / 20.50	9.73 / 48.50
75-25	FP32	88 / 749	30400.25 / 30070.31	386.03 / 38.50	345.36 / 39.93	2.89 / 24.91
75-25	INT8	334 / 1405	30148.58 / 30030.22	99.61 / 20.97	90.16 / 21.24	11.08 / 46.79
75-25	INT8_AC	296 / 1456	30125.39 / 30024.52	99.99 / 20.43	101.64 / 20.51	9.83 / 48.49
80-20	FP32	90 / 822	30385.87 / 30063.85	331.52 / 36.38	337.52 / 36.45	2.96 / 27.34
80-20	INT8	313 / 1422	30109.74 / 30015.86	97.27 / 20.90	96.10 / 20.98	10.40 / 47.37
80-20	INT8_AC	305 / 1316	30136.64 / 30032.92	97.68 / 21.96	98.71 / 22.64	10.12 / 43.82
85-15	FP32	96 / 821	30675.89 / 30066.39	340.82 / 36.42	319.43 / 36.49	3.13 / 27.31
85-15	INT8	296 / 1417	30168.85 / 30042.41	99.45 / 20.93	101.82 / 21.07	9.81 / 47.17
85-15	INT8_AC	300 / 1450	30152.14 / 30036.58	100.20 / 20.51	100.40 / 20.59	9.95 / 48.27

3.1. Impact of precision on performance

- INT8 models significantly outperformed FP32 models in terms of latency and throughput across both CPU and GPU.
- The reduction in model precision (from FP32 to INT8) led to an increase in throughput, with INT8 models achieving 3x–4x the FPS compared to FP32 on CPUs while ~2x on GPUs.
- INT8 (accuracy control) models exhibited slightly higher latency than standard INT8 models but maintained accuracy while still achieving ~3x speedup over FP32 on CPUs.
- For FP32 models, CPU achieved only ~3 FPS, while the GPU reached ~25 FPS, making GPU more suitable for real-time applications.

3.2. Dataset split and model efficiency

The dataset split had minimal impact on inference performance, as throughput and latency remained consistent across different training/testing distributions. This suggests that model conversion and precision tuning had a greater influence on performance than the dataset split.

3.3. Inference speed vs accuracy trade-off

- While INT8 models offer superior performance, their accuracy needs validation against application-specific requirements.
- The optimal balance for real-time queue management would be INT8 (accuracy control) on GPU, offering a 48+ FPS inference speed with minimal latency.

4. CONCLUSION

Future work could explore further optimizations, such as PTQ on additional hardware platforms and edge-device deployment to enhance accessibility in real-world applications. Additionally, adaptive quantization techniques could help balance accuracy loss while maximizing throughput. By leveraging OpenVINO optimizations, this work establishes a foundation for deploying AI-driven queue management systems in real time environments, ensuring efficiency and improved customer experience.

ACKNOWLEDGMENTS

We would like to thank the B.M.S College of Engineering for providing us with this wonderful opportunity. We would also like to thank the Department of Information Science and Engineering for their support and guidance.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

Rakshitha Mahendran contributed to the conceptualization, methodology, software development, validation, formal analysis, investigation, resource provision, data curation, original draft preparation, and review and editing. Sathwik Madhusudan contributed to methodology, validation, formal analysis, investigation, data curation, review and editing, visualization, supervision, and project administration. Omkar Prasad contributed to conceptualization, software development, validation, formal analysis, original draft preparation, visualization and review and editing. All authors reviewed and approved the final manuscript.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Chandrakala G. Raju	✓					✓			✓	✓			✓	
Ajaykumar Devarapalli		✓			✓	✓			✓	✓				
Rakshitha Mahendran	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓	
Sathwik Madhusudan		✓		✓	✓	✓		✓		✓	✓	✓	✓	
Omkar Prasad	✓		✓	✓	✓		✓		✓	✓	✓			

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

- coco dataset: http://images.cocodataset.org/zips/val2017.zip
- Video used for testing: https://github.com/openvinotoolkit/openvino_notebooks/blob/recipes/recipes/intelligent_queue_management/sample_video.mp4
- oneAPI documentation: https://www.intel.com/content/www/us/en/developer/tools/oneapi/overview.html#gs.k254pv
- Benchmarking tool: https://docs.openvino.ai/2025/get-started/learn-openvino/openvino-samples/benchmark-tool.html#advanced-usage-benchmark
- Benchmarking results: Benchmark_Table
- OpenVINO toolkit: https://www.intel.com/content/www/us/en/developer/tools/openvino-toolkit/overview.html

REFERENCES

[1] W. Liang, N. Fujita, R. Kobayashi and T. Boku, "Using SYCLomatic to migrate CUDA Code to oneAPI adapting NVIDIA GPU," *2024 IEEE International Conference on Cluster Computing Workshops (CLUSTER Workshops)*, Kobe, Japan, 2024, pp. 192-193, doi: 10.1109/CLUSTERWorkshops61563.2024.00054.

[2] H. Ahn *et al.*, "Performance characterization of using quantization for DNN inference on edge devices," *2023 IEEE 7th International Conference on Fog and Edge Computing (ICFEC)*, Bangalore, India, 2023, pp. 1-6, doi: 10.1109/ICFEC57925.2023.00009.

[3] A. Demidovskij *et al.*, "OpenVINO deep learning workbench: comprehensive analysis and tuning of neural networks inference," *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, Seoul, Korea (South), 2019, pp. 783-787, doi: 10.1109/ICCVW.2019.00104.

[4] S. Han, H. Mao, and W. J. Dally, "Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding," *ICLR 2016 (conference)*. Available: arXiv arXiv:1510.00149

[5] Y. Choi, M. El-Khamy, and J. Lee, "Towards the limit of network quantization," *arXiv preprint arXiv:1612.01543*, Dec. 2016 (revised Nov. 2017). DOI: 10.48550/arXiv.1612.01543.

[6] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," arXiv preprint arXiv:1603.05279, Mar. 2016, doi: 10.48550/arXiv.1603.05279.

[7] U. Kulkarni, A. S. Hosamani, A. S. Masur, S. Hegde, G. R. Vernekar and K. S. Chandana, "A survey on quantization methods for optimization of deep neural networks," *2022 International Conference on Automation, Computing and Renewable Systems (ICACRS)*, Pudukkottai, India, 2022, pp. 827-834, doi: 10.1109/ICACRS55517.2022.10028742.

[8] A. Mishra, E. Nurvitadhi, J. J. Cook, and D. Marr, "WRPN: wide reduced-precision networks," *arXiv preprint arXiv:1709.01134*, Sep. 2017, doi: 10.48550/arXiv.1709.01134.

[9] H. Wu, P. Judd, X. Zhang, and M. Isaev, "Integer quantization for deep learning inference: principles and empirical evaluation," *arXiv preprint arXiv:2004.09602*, Apr. 2020, doi: 10.48550/arXiv.2004.09602.

[10] Y. He, J. Lin, Z. Liu, H. Wang, L. J. Li, and S. Han, "AMC: AutoML for model compression and acceleration on mobile devices," in *Proceedings of the European Conference on Computer Vision*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham, Switzerland: Springer, 2018, pp. 815–832, doi: 10.1007/978-3-030-01234-2_48.

[11] C. Chen, Y. Gan, Z. Han, H. Gao, and A. Li, "An improved YOLOv5 detection algorithm with pruning and OpenVINO quantization," in *Proceedings 2023 China Automation Congress (CAC)*, Nov. 2023, pp. 1-6. doi: 10.1109/CAC59555.2023.10451021.

[12] M. Y. Chebtha, H. Ghilassi, and H. Chikh-Touami, "Real time drones detection based on cameras and recent YOLO algorithm versions," in *Proceedings 2023 2nd International Conference on Electronics, Energy and Measurement (IC2EM)*, Medea, Algeria, Nov. 2023, pp. 1–6. doi: 10.1109/IC2EM59347.2023.10419468.

[13] A. Demidovskij, A. Tugaryov, A. Kashchikhin, A. Suvorov, Y. Tarkan, F. Mikhail, and Y. Gorbachev, "OpenVINO deep learning Workbench: towards analytical platform for neural networks inference optimization," *Journal of Physics: Conference Series*, vol. 1828, no. 1, Art. no. 012012, Feb. 2021, doi: 10.1088/1742-6596/1828/1/01201.

[14] R. Dagli and S. Eken, "Deploying a smart queuing system on edge with Intel OpenVINO toolkit," *Soft Computing*, vol. 25, pp. 10103–10115, 2021, doi: 10.1007/s00500-021-05891-2.

[15] C. J. S. Schaefer, S. Joshi, S. Li, and R. Blazquez, "edge inference with fully differentiable quantized mixed precision neural networks," in *Proceedings IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2024, pp. 8460–8469.

[16] L. Kummer, K. Sidak, T. Reichmann, and W. Gansterer, "Adaptive precision training (AdaPT): a dynamic quantized training approach for DNNs," in *Proceedings SIAM International Conference on Data Mining (SDM)*, 2023, pp. 559–567, doi: 10.1137/1.9781611977653.ch6.

[17] M. Hussain, "YOLOv1 to v8: unveiling each variant–a comprehensive review of YOLO," in *IEEE Access*, vol. 12, pp. 42816-42833, 2024, doi: 10.1109/ACCESS.2024.3378568.

[18] Huang Yi, Sun Shiyu, Duan Xiusheng and Chen Zhigang, "A study on deep neural networks framework," *2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, Xi'an, China, 2016, pp. 1519-1522, doi: 10.1109/IMCEC.2016.7867471.

[19] C. Liqun and H. Lei, "Clipping-based neural network post training quantization for object detection," *2023 IEEE International Conference on Control, Electronics and Computer Technology (ICCECT)*, Jilin, China, 2023, pp. 1192-1196, doi: 10.1109/ICCECT57938.2023.10141287.

[20] D. Przewlocka-Rus, T. Kryjak and M. Gorgon, "PowerYOLO: mixed precision model for hardware efficient object detection with event data," *2024 27th Euromicro Conference on Digital System Design (DSD)*, Paris, France, 2024, pp. 210-217, doi: 10.1109/DSD64264.2024.00036.

[21] E. Y. Pradana, S. A. Aji, M. A. Abdulrrozaq, A. H. Alasiry, A. Risnumawan and E. Pitowarno, "Optimizing YOLOv8 for real-time performance in humanoid soccer robots with OpenVINO," *2024 International Electronics Symposium (IES)*, Denpasar, Indonesia, 2024, pp. 304-309, doi: 10.1109/IES63037.2024.10665829.

[22] N. Andriyanov and G. Papakostas, "Optimization and benchmarking of convolutional networks with quantization and OpenVINO in Baggage image recognition," *2022 VIII International Conference on Information Technology and Nanotechnology (ITNT)*, Samara, Russian Federation, 2022, pp. 1-4, doi: 10.1109/ITNT55410.2022.9848757.

[23] W. Perera, H. Perera, H. Perera, S. Suraweera, S. Rathnayake and S. Karunathilaka, "Dynamic queue management system for multi-cashier supermarkets using IoT and real-time monitoring," *2024 6th International Conference on Advancements in Computing (ICAC)*, Colombo, Sri Lanka, 2024, pp. 169-174, doi: 10.1109/ICAC64487.2024.10850938.

[24] S. S. Kadam, T. Jadhav, L. Patil, P. Saw and A. Landage, "Crowd counting using yolov8 and various tracking algorithms," *2024 OPJU International Technology Conference (OTCON) on Smart Computing for Innovation and Advancement in Industry 4.0*, Raigarh, India, 2024, pp. 1-9, doi: 10.1109/OTCON60325.2024.10688023.

[25] A. Elaoua, M. Nadour, L. Cherroun and A. Elasri, "Real-time people counting system using YOLOv8 Object Detection," *2023 2nd International Conference on Electronics, Energy and Measurement (IC2EM)*, Medea, Algeria, 2023, pp. 1-5, doi: 10.1109/IC2EM59347.2023.10419684.

BIOGRAPHIES OF AUTHORS

Chandrakala G. Raju is an assistant professor in the Department of Information Science at B.M.S. College of Engineering, Bengaluru, Karnataka, India. She holds a Ph.D. in computer science and engineering and has a research focus on data mining, machine learning, and artificial intelligence. She has published several research papers in reputed journals and conferences. She is a member of professional bodies such as the IEEE and ACM, and she can be contacted at email: chandrakalagraju.ise@bmsce.ac.in.

Ajaykumar Devarapalli received his B.Tech. (electronics and communications engineering) from JNTU, Hyderabad, Andhra Pradesh, and M.Tech. (integrated circuit technology) from University of Hyderabad, Hyderabad, Andhra Pradesh. He is persuing Ph.D. degree from Department of Electrical and Electronics Engineering, National Institute of Technology Karnataka, Surathkal, India. He has 2 years of experience in the industry and 15 years in teaching. He is an active IEEE member and has published 5 national and 5 International Journal Papers, Filed 03 Indian Patents. He has been a reviewer, program committee member, coordinator, organizer and guest speaker for technical conferences/contests/workshops in the domain of VLSI and SoC. He can be contacted at email: d.ajay402@gmail.com.

Rakshitha Mahendran is a final year student pursuing her information science and engineering in B.M.S. College of Engineering, Bengaluru, Karnataka, India. She was a part of the marketing committee for the international symposium PhaseShift 2022 as a junior coordinator. She has participated in multiple codeathons and hackathons at the college level. She is passionate about cloud computing and networks in the software domain. She can be contacted at email: rakshitha4210@gmail.com.

Sathwik Madhusudan is a final-year information science engineering student at BMS College of Engineering. He is passionate about AI and machine learning, aspiring to integrate these technologies into various aspects of life. His research focuses on fine-tuning open-source large language models and exploring their applications in real-world problem-solving. Beyond academics, he has worked on projects in cybersecurity, DevOps, and smart waste management. He is also an active volunteer at U&i, mentoring students and contributing to social impact initiatives. Soon, he will begin his professional career at Hewlett Packard Enterprise, where he aims to further his expertise and drive innovation in AI and cloud computing. He can be contacted at email: sathwikmadhusudan@gmail.com.

Omkar Prasad is currently pursuing his final year B.E. in information science and engineering at BMS College of Engineering, Bengaluru, Karnataka. He is an active IEEE member and has participated in various IEEE activities. His research work on quantum entanglement has been published showcasing his keen interest in scientific exploration and innovation. And he has participated in various coding competitions including codeathons and hackathons. He can be contacted at email: omkaar@gmail.com.