Comparative Analysis of Lightweight Vision Models on Embedded Accelerator Devices

Greta Corti[1], Christian Veronesi[1], Pietro Bartoli[1], Sara Pidò[1], Andrea Giudici[1], Francesca Palermo[2], Diana Trojaniello[2], Matteo Matteucci[1], and Simone Mentasti[1]

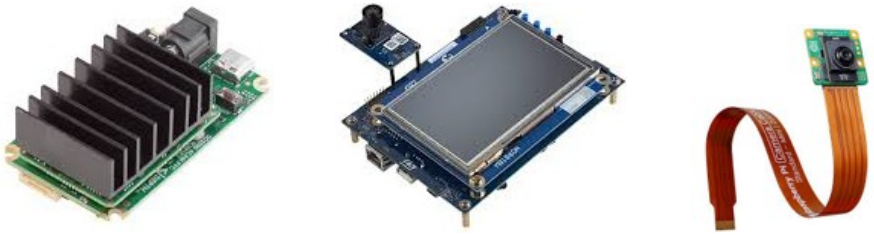
Politecnico di Milano, Milan 20133, Italy, {name}.{surname}@polimi.it
EssilorLuxottica, Milan 20133, Italy, {name}.{surname}@luxottica.com

Abstract. The rapid growth of edge devices demands efficient neural network accelerators for low-latency, energy-efficient AI tasks like object detection and pose estimation. Despite benefits such as improved privacy and reduced cloud dependence, edge AI faces challenges including hardware heterogeneity, resource constraints, and the balance between real-time performance and accuracy. This paper benchmarks STMicroelectronics STM32N6, Luxonis OAK-FCC 4P, and SONY IMX500 on object detection and human pose estimation tasks, evaluating power consumption, inference time, accuracy, and memory usage. Using YOLOv5 and YOLOv8 models in Small and Nano configurations on COCO and COCO-Pose datasets, the study highlights the trade-offs between efficiency and accuracy. Results show IMX500 leads in inference time and power efficiency but is limited in memory compared to N6 and OAK-FCC 4P. This study provides a comprehensive evaluation of the suitability of different hardware architectures, guiding the optimization of lightweight neural network deployment on resource-constrained platforms.

Keywords: Lightweight Neural Network · Neural Network Accelerators · Hardware Benchmarking · Low-Power Edge Devices

1 Introduction

The increasing availability of edge devices, e.g., IoT sensors, mobile phones, drones, has driven the demand for low-latency, efficient neural network accelerators capable of enabling real-time AI tasks such as object detection and pose estimation [14,38], reducing reliance on cloud computing [37] and addressing privacy concerns [20]. However, deploying AI on the edge introduces challenges, notably hardware heterogeneity, limited memory, restricted power budgets, and the difficulty of achieving high accuracy under strict resource constraints [10,24]. Selecting an appropriate neural accelerator thus requires careful consideration of these trade-offs. For example, certain edge devices, such as the Luxonis OAK-FCC 4P board [22], shown in Figure 1a, and STMicroelectronics STM32N6 [32], shown in Figure 1b, are designed to facilitate efficient processing and minimize power consumption through the incorporation of dedicated neural compute engines. These devices are designed to handle real-time inference for applications



(a) Luxonis OAK-FCC 4P (b) STMicroelectronics STM32N6 (c) SONY IMX500

Fig. 1: Target devices selected for benchmarking analysis.

such as object detection and pose estimation while operating within strict power budgets. In contrast, other devices prioritize on-chip inference, integrating AI capabilities directly into sensors or compact processors with highly constrained memory footprints, such as the SONY IMX500 camera [30], shown in Figure 1c.

While the hardware landscape evolves rapidly, there is still limited research exploring their potential and performances of these platforms on state-of-the-art neural network architectures. This highlights a critical gap where further investigation is needed to advance the field and enable more effective and efficient AI deployment on edge devices. To address this, we benchmark three representative edge devices, STM32N6, OAK-FCC 4P, and IMX500, on object detection and human pose estimation using You Only Look Once (YOLO) models (5 and 8 versions, Nano and Small variants), trained on COCO and COCO-Pose datasets. Key contributions of this paper are:

- Benchmarking accuracy, inference speed, memory usage, and power consumption across the three devices using YOLO.
- Providing an in-depth analysis of quantization challenges in YOLO models and proposing possible solutions to address them.
- Providing insights for optimizing lightweight neural networks for deployment on resource-constrained platforms.

2 Related Works

Recent advancements in neural network accelerators have enabled real-time inference on resource-constrained devices, playing a key role in edge computing, IoT, and autonomous systems [9,24]. Efficient hardware-software co-design has become critical to balancing latency, energy consumption, and accuracy. AI accelerators can be implemented in several forms:

- **Dedicated ASICs**: Specialized chips such as Google's Edge TPU [11] and Intel's Movidius Myriad X [15] deliver efficient AI inference.
- **FPGAs**: Configurable circuits that can be repeatedly programmed after manufacturing.
- **Embedded GPUs**: General-purpose processors (e.g., NVIDIA Jetson [26]) able to perform mathematical calculations at high speeds in parallel.

These accelerators are optimized for the execution of parallel AI workloads, including matrix multiplication and convolution, which are prevalent operations

Table 1: Specifications of the Target Edge Devices
(-: values are not available)

Specification	OAK-FCC 4P	STM32N6	IMX500
Memory Available (MB)	323	64	8
TOPS	1.4	0.6	-
Max Power Consumption (W)	4.5	3	3
W/TOPS	3.2	5	-
Supported Frameworks	Caffe, ONNX, TensorFlow	ONNX, TensorFlow	PyTorch, TensorFlow

in deep learning models. The market offers a variety of artificial intelligence accelerators, each with distinct features and performance characteristics. These accelerators differ in their processing capabilities, power consumption, and supported AI frameworks, underscoring the importance of selecting an appropriate accelerator to align with the specific requirements of the edge AI application.

System-on-Chip Application-Specific Integrated Circuits (ASICs) are specialized chips designed to perform dedicated tasks, e.g. digital recording [7], video compression [25], neural network inference [18]. Modern ASICs integrate microprocessors, memory, and peripherals, earning the term System-on-Chip (SoC).

Nowadays, there has been a growing trend in energy-efficient SoC designs for real-time neural network models. Among these, Intel's Movidius Myriad X Vision Processing Unit (VPU) [29,10] remains a popular benchmark, although its performance for lightweight object detection and pose estimation remains underexplored relative to newer platforms like STM32N6 and IMX500.

Lightweight Deep Neural Networks Deep neural networks, powered by GPU acceleration, have achieved state-of-the-art performance in object detection and pose estimation [6,28,34,8,33]. Models such as YOLOv4 [6], YOLOv5 [28], and EfficientDet [34] for object detection, as well as OpenPose [8] and HRNet [33] for human pose estimation, have exhibited remarkable accuracy and speed when executed on high-performance GPU platforms. However, their high computational and memory demands limit their deployment on edge devices. To address this, lightweight networks such as YOLO Nano and YOLO Small offer significant reductions in model size and complexity, enabling real-time inference at the edge. This decrease in parameters, however, is accompanied by a corresponding decline in accuracy. YOLO Nano models reduce parameters by 80–85% compared to Medium configurations, with an associated 5–15% accuracy drop; Small models achieve a 30–40% reduction with a smaller accuracy loss of 3–8%. These models are designed to balance the reduction of accuracy with the acceleration of inference and the reduction of memory usage.

3 Methodology

In this section, we first describe the neural network models utilized in our experiments for edge device comparison. We then compare different edge device architectures, summarized in Table 1.

3.1 YOLO Detection and Pose Models

To evaluate the performance of various accelerators, we employed two key computer vision tasks: object detection and pose estimation, both essential for edge computing where efficiency and accuracy are critical.

YOLO was chosen as the reference model due to its state-of-the-art performance and broad adoption. We used the PyTorch Ultralytics implementation [16] for object detection (YOLOv5 and YOLOv8, Nano and Small versions) and for human pose estimation (YOLOv8 Nano and Small).

Although newer versions of YOLO (e.g., v11, v12) have been recently released, YOLOv5 and YOLOv8 were preferred due to their extensive validation, stable performance, and widespread support in real-world applications. Furthermore, they offer a balanced trade-off between model complexity and hardware compatibility, which is crucial for edge deployment.

The models were resized from 640×640 pixels to 224×224 to fit the memory constraints of the target devices. All models were retrained from the original weights using Adam optimizer [17] with default hyperparameters. Object detection models were trained using COCO (Common Objects in Context) dataset [19], while pose estimation models were trained on a specialized version, COCO-Pose dataset. All models were successfully deployed, although Non-Maximum Suppression was performed on a PC due to hardware limitations.

3.2 Edge AI Hardware Platforms Deployment

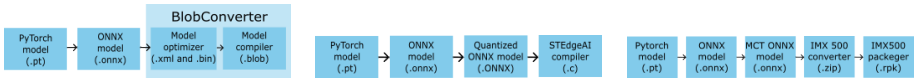
The following section provides a comprehensive analysis of the hardware components of the three devices considered in our study: OAK-FCC 4P, STM32N6, and IMX500. Additionally, we present an overview of the deployment pipelines used for the models on each edge device.

Luxonis' OAK-FCC 4P The OAK-FFC 4P, designed by Luxonis, is a versatile prototyping platform that supports interconnection with various camera modules. It integrates the Movidius Myriad X SoC inside the Robotics Vision Core 2 [23], enabling neural network inference directly on the device without external computation. The board features approximately 4.5 W maximum power consumption, 1.4 TOPS AI capability, and around 323 MB available memory.

The Myriad X VPU includes Intel's Neural Compute Engine, supporting Floating Point (FP) 16 and 8-bit fixed-point operations, and enabling deployment of networks in Caffe [2], TensorFlow [35], and ONNX [27] formats.

Since PyTorch models are not natively supported, a dedicated deployment pipeline was required. Models were first converted to ONNX format using PyTorch tracing, then transformed into a `.blob` file using Luxonis' BlobConverter [1]. BlobConverter optimizes the model into OpenVINO IR format and compiles it into a final BLOB file. The `.blob` was executed on the device using Luxonis' DepthAI Python API [21]. The full pipeline is illustrated in Figure 2a.

STMicroelectronics' STM32N6 The STM32N6 (N6) is a Microcontroller Unit (MCU) developed by ST-Microelectronics using 16 nm FFC technology. Designed for energy-efficient AI and ML workloads in battery-powered devices, it features an ARM Cortex-M55 CPU operating up to 800 MHz with vector processing, TrustZone, and Arm Helium extensions for DSP and ML efficiency [4]. In addition to its CPU, the N6 integrates ATON, a high-performance CNN accelerator comprising convolution, scalar, activation, pooling, and decompression



(a) OAK-FCC 4P (b) STM32N6 (c) IMX500

Fig. 2: Target devices deployment pipelines.

units, along with a reconfigurable buffer and high-throughput AXI interfaces. An epoch controller further streamlines configuration and reduces CPU load.

As shown in Figure 2b, the deployment pipeline is similar to that of the OAK-FCC 4P. Models are converted from PyTorch to ONNX, then statically quantized to 8-bit precision using the ONNX Runtime library [3] by inserting QuantizeLinear and DeQuantizeLinear operators. The quantized model is subsequently compiled using ST's STEdgeAI tool, producing a `.c` file for deployment and an AI runtime model to verify whether the model was compiled optimally.

SONY IMX500 The IMX500, developed by SONY, is the first Intelligent Vision Sensor with integrated edge processing. It features a stacked structure combining an image sensor, high-performance DSP, and on-chip static RAM, enabling high-speed AI inference without remote server transmission.

In this study, we used the IMX500 integrated with a Raspberry Pi, following SONY's documentation for specifications and deployment. While the processor's TOPS are unspecified, its power consumption remains below 3 W, with an available memory of 8 MB. The IMX500 supports models in both TensorFlow and PyTorch formats. For deployment, PyTorch models were quantized and compressed using Sony's Model Compression Toolkit (MCT) [31] into ONNX format, then converted into binary files via the IMX500 Converter [12]. The resulting `.zip` file includes a memory report and is packaged into an RPK file using the IMX500 Packager [13], then deployed to the Raspberry Pi camera through a Python application. The complete pipeline is shown in Figure 2c.

4 Experiments and Results

In this section, we compare the available memory, model utilization, and hardware limitations of the three edge devices introduced in 3.2, followed by an evaluation of inference speed for real-time performance. Next, we assess post-deployment accuracy and possible degradation. Finally, we examine power consumption to evaluate energy efficiency for low-power applications.

4.1 Memory Usage

As shown in 3.2 and 1, memory varies across devices: OAK-FCC 4P supports up to 323 MB, STM32N6 up to 64 MB, and IMX500 up to 8 MB.

2 shows the memory footprint of YOLO models for object detection and human pose estimation across the devices. While the initial model sizes are identical, the deployed memory usage differs due to device-specific compiler optimizations and memory management strategies. Notably, Nano models occupy about 50% of the IMX500's memory, but only around 6% on STM32N6 and 3%

Table 2: Memory usage (MB) of object detection and pose estimation models evaluated on target devices. ((E): the model exceeds the available memory of the device; -: the model is not currently capable of being deployed on the device.)

Model	OAK-FCC 4P ↓	STM32N6 ↓	IMX500 ↓
Object detection models			
YOLO v5 n	7.07	**3.86**	-
YOLO v5 s	22.54	**10.80**	-
YOLO v8 n	8.02	4.35	**4.05**
YOLO v8 s	26.38	**12.78**	11.87 (E)
Human pose estimation models			
YOLO v8 n	8.14	4.41	**4.15**
YOLO v8 s	27.15	**13.24**	12.27 (E)

Table 3: Inference times (ms) of object detection and human pose estimation models evaluated on target devices.

Model	OAK-FCC 4P ↓	STM32N6 ↓	IMX500 ↓
Object detection models			
YOLO v5 n	22.04	**21.14**	-
YOLO v5 s	**37.67**	47.84	-
YOLO v8 n	21.39	23.44	**5.55**
YOLO v8 s	**40.62**	54.38	-
Human pose estimation models			
YOLO v8 n	22.56	24.39	**5.84**
YOLO v8 s	**43.25**	57.64	-

on OAK-FCC 4P. The larger footprint on OAK-FCC 4P is due to its compiler using FP16 precision, whereas STM32N6 and IMX500 employs int8 quantization. Overall, the IMX500's limited memory significantly restricts its ability to deploy larger models, unlike STM32N6 and OAK-FCC 4P, which comfortably handle more complex networks. With a maximum of 8 MB of memory, the IMX500 has already reached 50% of its capacity with YOLO Nano models and this limits its suitability for deployment of larger models on this hardware. In contrast, the other two devices can handle larger models with greater ease.

4.2 Inference Time

Given an input image, we compared the inference times by measuring how long each device takes to produce a prediction for the estimation models. This allows for an evaluation on their performance and an understanding of their suitability for running neural models in real-time, key objective for edge accelerators.

Each device is able to measure this value internally and reporting it to the user. The inference times, in 3, are the results of the arithmetic mean of inferences performed on images from the COCO dataset for object detection and the COCO-Pose dataset for human pose estimation.

The values calculated on the OAK-FCC 4P and the STM32N6 are highly similar, differing by only a few milliseconds for the Nano models of both object detection and human pose estimation. Both devices are capable of running the models in real-time, achieving an inference rate of approximately 45 Hz.

The YOLO v8 Nano model on IMX500 demonstrates superior performance in terms of processing speed and frequency when compared to the other two devices, operating at approximately 200 Hz.

Table 4: Accuracy results considering mAP@[0.50:0.95] in percentage of the two models after conversions into ONNX and quantized ONNX.

Model	PyTorch ↑	ONNX ↑	Quantized ONNX ↑
Object detection models			
YOLO v5 n	22.0	21.8	**10.6**
YOLO v5 s	28.4	28.2	3.07
YOLO v8 n	23.6	23.4	3.5
YOLO v8 s	**29.3**	**29.1**	7.9
Human pose estimation models			
YOLO v8 n	26.6	26.6	3.1
YOLO v8 s	**35.9**	**35.8**	**7.2**

Table 5: Accuracy results considering mAP@[0.50:0.95] in percentage of the two models considering the three solutions for the quantization problem.

Model	ONNX ↑	ONNX norm bbox ↑	Mixed quant ONNX ↑	Quant ONNX norm bbox ↑	Quant ONNX multiple outputs ↑	MCT quant ONNX multiple outputs ↑
Object detection models						
YOLO v5 n	21.8	21.5	19.9	17.8	17.2	**20.0**
YOLO v5 s	28.2	27.9	**16.4**	9.9	10.2	-
YOLO v8 n	23.4	23.2	14.2	8.6	12.5	**22.0**
YOLO v8 s	29.1	28.7	**17.0**	13.4	13.1	-
Human pose estimation models						
YOLO v8 n	26.6	26.1	**15.8**	4.9	14.1	15.6
YOLO v8 s	35.8	35.3	**22.3**	9.2	20.0	-

4.3 Accuracy Evaluation

During training, model weights and activations are optimized in FP32, whereas edge accelerators typically operate with lower-precision formats such as integers. Accurate deployment required the pipelines described in 3.2. To assess model performance during the deployment process, we measured accuracy after each conversion step. As reference metric, we used mean Average Precision with Intersection over Union (IoU) thresholds from 0.50 to 0.95 (mAP@[0.50:0.95]) over the 80 COCO object detection classes and 17 COCO-Pose keypoints.

As discussed in 3.1, all models were retrained using resized inputs due to hardware memory limitations. Table 4 shows that conversion to ONNX format preserves accuracy, while quantization with ONNX Runtime causes a significant drop of at least 70%. Further investigation revealed that the degradation was caused by the structure of YOLO model heads. Bounding box coordinates (ranging from 0 to image size) and confidence scores (ranging from 0 to 1) are quantized with different scale factors, leading to instability and accuracy loss.

To address the quantization issue, three potential solutions were identified:

- Exclude the final head layers from quantization, keeping them in FP32;
- Remove the final concatenation, separately quantize bounding box and score outputs;
- Normalize bounding box coordinates relative to the input image size, bringing them into the [0,1] range to stabilize full quantization.

Table 5 summarizes the results of these approaches. The "ONNX norm bbox" column shows accuracy after normalization, which causes a slight decline due to internal model changes. The "MCT quant ONNX multiple outputs" column reports results using Sony's MCT tool, while the other columns correspond to ONNX Runtime quantization methods. Compared to Table 4, all strategies significantly improve quantized model accuracy, bringing it closer to the original ONNX results. Among them, MCT achieves the highest overall accuracy due to the internally implemented methods. Particularly, the MCT framework applies graph optimizations, like BatchNorm folding, during calibration collects activation statistics for estimating the quantization thresholds for each layers and finally it uses symmetric quantization with power-of-two thresholds. These steps help improve accuracy compared to ONNXRuntime, which performs standard PTQ without advanced graph transformations or threshold tuning. Mixed

Table 6: Accuracy results considering mAP@[0.50:0.95] in percentage of the two models on OAK-FCC 4P and STM32N6 on the COCO dataset.

Model	OAK-FCC 4P ↑	STM32N6 norm bbox ↑	STM32N6 multiple outputs ↑
Object detection models			
YOLO v5 n	**21.8**	17.8	17.2
YOLO v5 s	**28.2**	9.9	10.2
YOLO v8 n	**23.4**	8.6	12.5
YOLO v8 s	**29.1**	13.4	13.1
Human pose estimation models			
YOLO v8 n	**26.6**	4.9	14.1
YOLO v8 s	**35.7**	9.2	20.0

Table 7: Accuracy results considering mAP@[0.50:0.95] in percentage of the two models with multiple outputs on OAK-FCC 4P, STM32N6, and IMX500 on the custom datasets captured by IMX500.

Model	OAK-FCC 4P ↑	STM32N6 ↑	IMX500 ↑
Object detection model			
YOLO v8 n	**20.3**	10.8	19.5
Human pose estimation model			
YOLO v8 n	**32.8**	21.2	22.4

quantization via ONNX Runtime also performs well but is unsuitable for deployment, as the partially floating-point head cannot run on accelerators. Therefore, for STM32N6 deployment, full quantization strategies were preferred despite slightly lower accuracy. The OAK-FCC 4P did not suffer from this issue, as its compiler maintains high accuracy after quantization, as shown in Table 6. In contrast, STM32N6 models exhibited noticeable degradation. However, analysis shows the accuracy loss originates from the quantization process, not from ST's compiler. We observed a significant drop in accuracy for the YOLOv5s model when using ONNXRuntime PTQ, which might be related to numerical instabilities introduced during the quantization process. A deeper analysis of the quantization methods and their impact on robustness is planned as future work.

IMX500 results are excluded here, as COCO and COCO-Pose datasets could not be directly transmitted to the Raspberry Pi AI Camera during experiments. A dedicated validation setup was created for this device: dataset images were displayed on a screen while a script allowed users to navigate images, capture them through the IMX500 to account for camera distortions, run inference, and compute accuracy. Direct comparison with Table 6 is not feasible due to slight perspective changes. Therefore, models were re-evaluated on OAK-FCC 4P and STM32N6 using the IMX500-captured dataset. Table 7 presents the results.

For object detection, OAK-FCC 4P and IMX500 show similar performance, whereas STM32N6 exhibits about half the accuracy, mainly due to the quantization from FP32 to integer. As shown in Figure 3, N6 predictions display many errors, while OAK-FCC 4P and IMX500 maintain reliable detections.

For human pose estimation, the gap narrows: STM32N6 and IMX500 achieve comparable accuracy, although OAK-FCC 4P remains superior with a mAP of 32.8%. As illustrated in Figure 4, OAK-FCC 4P predictions closely match ground truth, while the other devices show greater keypoint localization errors.

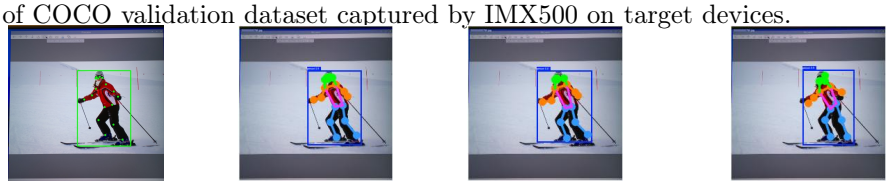
Table 6 and Table 7 further demonstrate that screen characteristics do not impact the experiments. Indeed, when evaluating the same device on the same dataset but with different cameras, we observe that the accuracy stays consistent.

4.4 Power Consumption

Power consumption is crucial for battery-powered devices, as excessive energy usage can compromise integration in IoT applications. Edge AI devices often

(a) Ground Truth (b) OAK-FCC 4P (c) STM32N6 (d) IMX500

Fig. 3: Qualitative comparison of object detection predictions on the same image of COCO validation dataset captured by IMX500 on target devices.



(a) Ground Truth (b) OAK-FCC 4P (c) STM32N6 (d) IMX500

Fig. 4: Qualitative comparison of pose estimation predictions on the same image of COCO-Pose validation dataset captured by IMX500 on target devices.

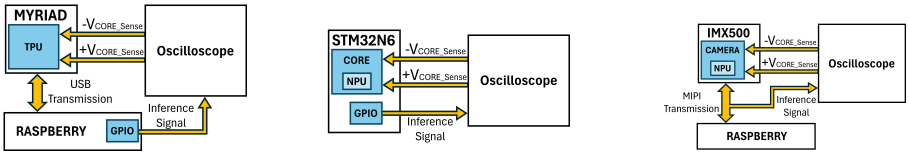
show significant power variability due to differences in data paths, preprocessing, peripherals, and firmware, complicating cross-device comparisons [5].

This section analyzes the power usage of the devices listed in Table 1. To accurately measure inference energy, a GPIO pin was raised at the start and lowered at the end of inference (when supported), defining a precise measurement window and minimizing signal transition effects [36]. While rise and fall times were negligible, each device required specific ad-hoc adaptations to enable reliable power measurements.

IMX500 The IMX500 camera core's energy consumption was measured, but firmware limitations prevented using a GPIO signal to isolate inference. Instead, a MIPI signal, which goes high when the camera is idle, monitored activity. A 220 $m\Omega$ shunt resistor was inserted into the core's supply path to enable precise oscilloscope measurements. Figure 5c shows the setup diagram.

OAK-FCC 4P For the OAK-FCC 4P, the inference phase was isolated using a Raspberry Pi GPIO pin, which was set high before inference and reset low afterward, according to the block diagram reported in Figure 5a. This was enabled by modifying the device management script, and the timing accuracy was verified against the device's logs. To measure energy consumption, a 33 $m\Omega$ resistor was added in series with the core after cutting the power supply trace, following the same approach used for the IMX500.

STM32N6 For the STM32N6, the energy consumption of the core was measured using a GPIO to isolate the inference phase exclusively. The core voltage was fixed at 900 mV, with clock frequencies set to 1 GHz for the NPU, 800 MHz for the CPU, 400 MHz for the system RAMs, and 900 MHz for the NPU RAMs.



(a) OAK-FCC 4P (b) STM32N6 (c) IMX500

Fig. 5: Block diagrams of the measurement setup used for target devices.

Table 8: Energy consumption of the considered target devices for object detection and human pose estimation models (values in mJ).

Model	OAK-FCC 4P ↓	STM32N6 ↓	IMX500 ↓
Object detection models			
YOLO v5 n	30.5	**6.23**	-
YOLO v5 s	49.1	**16.62**	-
YOLO v8 n	30.22	7.33	**4.5**
YOLO v8 s	54.03	**18.06**	-
Human pose estimation models			
YOLO v8 n	30.14	7.13	**4.7**
YOLO v8 s	58.26	**19.78**	-

To evaluate the performance of the N6, the STM32N6 Discovery kit was configured following the block diagram reported in Figure 5b. Leveraging these configurations, the power consumption can be calculated as $P_{\text{core}}(t) = I_{\text{core}}(t) \cdot V_{\text{core}}$ where I_{core} and V_{core} are the core current and the core voltage, respectively. Since the inference execution time is marked by the trigger signal, the total energy consumption during inference is computed as $E_{core} = \int_{t_0}^{t_1} P_{\text{core}}(t) \cdot dt$ where t_0 and t_1 represent the start and end of the trigger signal, respectively, E_{core} represents the total energy consumed by the core over this period. Table 8 shows the measured mJ when different versions of YOLO models are executed.

5 Conclusion and Future Works

This paper presented a benchmarking analysis of lightweight neural network models, specifically YOLO Nano and Small, on three hardware accelerators: OAK-FCC 4P, STM32N6, and IMX500. We evaluated key metrics including accuracy, inference speed, memory usage, and power consumption, highlighting the trade-offs of deploying models on edge devices. The IMX500's primary limitation is its constrained memory, restricting the use of YOLO Small models, while STM32N6 and OAK-FCC 4P can accommodate larger networks. In inference speed, IMX500 outperforms the others, while STM32N6 and OAK-FCC 4P deliver real-time performance on Nano models. The OAK-FCC 4P achieves the highest accuracy, although quantization introduced some degradation. In terms of energy consumption, IMX500 is the most efficient, followed closely by STM32N6. Our experiments highlight the need for more robust post-training quantization tools for the N6 platform, as current ONNXRuntime PTQ methods can significantly impact model accuracy, especially for complex architectures. Future work will focus on improving quantization techniques to exploit hardware capabilities and expanding the benchmarking to new emerging accelerators.

Acknowledgments

This work was carried out in the Smart Eyewear Lab, a Joint Research Center between EssilorLuxottica and Politecnico di Milano.

References

1. Blobconverter: Convert models for luxonis devices. https://blobconverter.luxonis.com/
2. Caffe. https://caffe.berkeleyvision.org/
3. GitHub - microsoft/onnxruntime: ONNX Runtime: cross-platform, high performance ML inferencing and training accelerator — github.com. https://github.com/microsoft/onnxruntime
4. arm: Arm® CoreSight™ ETM-M55 Technical Reference Manual
5. Banbury, C., Reddi, V.J., Peter Torelli, e.a.: Mlperf tiny benchmark (2021)
6. Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934 (2020)
7. Byun, K.J., Hahn, M., Kim, K.S.: Implementation of 13 kbps qcelp vocoder asic. In: AP-ASIC'99. First IEEE Asia Pacific Conference on ASICs (Cat. No. 99EX360). pp. 258–261. IEEE (1999)
8. Cao, Z., Simon, T., Wei, S.E., Sheikh, Y.: Realtime multi-person 2d pose estimation using part affinity fields. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7291–7299 (2017)
9. Carrión, D.S., Prohaska, V.: Exploration of tpus for ai applications. arXiv preprint arXiv:2309.08918 (2023)
10. Dunkel, E., et al.: Benchmarking deep learning inference of remote sensing imagery on the qualcomm snapdragon and intel movidius myriad x processors onboard the international space station. In: IGARSS 2022 - IEEE International Geoscience and Remote Sensing Symposium. pp. 5301–5304. Kuala Lumpur, Malaysia (2022)
11. Google: Introduction to cloud tpu | google cloud. https://cloud.google.com/tpu/docs/intro-to-tpu
12. Group, S.S.S.: Imx500 converter: Aitrios. https://developer.aitrios.sony-semicon.com/en/raspberrypi-ai-camera/documentation/imx500-converter?version=3.14.3&progLang=, accessed: 2024-12-11
13. Group, S.S.S.: Imx500 packager: Aitrios. https://developer.aitrios.sony-semicon.com/en/raspberrypi-ai-camera/documentation/imx500-packager?version=2024-11-21&progLang=, accessed: 2024-12-11
14. Humes, E., Navardi, M., Mohsenin, T.: Squeezed edge yolo: Onboard object detection on edge devices. arXiv preprint arXiv:2312.11716 (2023)
15. Intel: Intel® movidius™ myriad™ x vision processing unit (vpu). https://www.intel.com/content/www/us/en/products/details/processors/movidius-vpu/movidius-myriad-x/products.html
16. Jocher, G., Qiu, J., Chaurasia, A.: Ultralytics yolo (version 8.0.0) [computer software]. https://github.com/ultralytics/ultralytics
17. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
18. Lee, S.S., Nguyen, T.D., Meher, P.K., Park, S.Y.: Energy-efficient high-speed asic implementation of convolutional neural network using novel reduced critical-path design. IEEE Access **10**, 34032–34045 (2022)

19. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common Objects in Context. In: Computer Vision – ECCV 2014. Lecture Notes in Computer Science, vol. 8693, pp. 740–755. Springer International Publishing (2014)
20. Lodge, T., Crabtree, A., Brown, A.: Developing gdpr compliant apps for the edge. In: International Workshop on Data Privacy Management. Springer (2018)
21. Luxonis: Luxonis/depthai-python: Depthai python library. https://github.com/luxonis/depthai-python
22. Luxonis: OAK-FFC 4P. https://docs.luxonis.com/hardware/products/OAK-FFC%204P
23. Luxonis: Robotics vision core 2 (rvc2). https://docs.luxonis.com/hardware/platform/rvc/rvc2/#RVC2%20NN%20Performance
24. Murshed, M.S., Murphy, C., Hou, D., Khan, N., Ananthanarayanan, G., Hussain, F.: Machine learning at the network edge: A survey. ACM Computing Surveys (CSUR) **54**(8), 1–37 (2021)
25. Núñez-Yáñez, J.L., Chouliaras, V.A.: Design and implementation of a high-performance and silicon efficient arithmetic coding accelerator for the h. 264 advanced video codec. In: 2005 IEEE International Conference on Application-Specific Systems, Architecture Processors (ASAP). pp. 411–416. IEEE (2005)
26. NVIDIA: Nvidia embedded systems for next-gen autonomous machines. www.nvidia.com/en-in/autonomous-machines/embedded-systems/
27. ONNX: Open neural network exchange. https://onnx.ai/
28. Redmon, J.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition (2016)
29. Reuther, A., Michaleas, P., Jones, M., Gadepally, V., Samsi, S., Kepner, J.: Survey and benchmarking of machine learning accelerators. In: 2019 IEEE High Performance Extreme Computing Conference (HPEC). pp. 1–9. Waltham, MA, USA
30. Sony: IMX500. https://developer.sony.com/imx500
31. Sony: Sony/model_optimization: Model compression toolkit (mct). https://github.com/sony/model_optimization, accessed: 2024-12-11
32. STMicroelectronics: STM32N6: Get a sneak peek at the future of AI-powered MCUs. https://blog.st.com/stm32n6/
33. Sun, K., Xiao, B., Liu, D., Wang, J.: Deep high-resolution representation learning for human pose estimation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 5693–5703 (2019)
34. Tan, M., Pang, R., Le, Q.V.: Efficientdet: Scalable and efficient object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10781–10790 (2020)
35. TensorFlow: Tensorflow. https://www.tensorflow.org/
36. Tschand, A., Rajan, A.T.R., Idgunji, S., Ghosh, A., Holleman, J., Kiraly, C., Ambalkar, P., Borkar, R., Chukka, R., Cockrell, T., Curtis, O., Fursin, G., Hodak, M., Kassa, H., Lokhmotov, A., Miskovic, D., Pan, Y., Manmathan, M.P., Raymond, L., John, T.S., Suresh, A., Taubitz, R., Zhan, S., Wasson, S., Kanter, D., Reddi, V.J.: Mlperf power: Benchmarking the energy efficiency of machine learning systems from microwatts to megawatts for sustainable ai (2024)
37. Tsoukas, V., Gkogkidis, A., Boumpa, E., Kakarountas, A.: A review on the emerging technology of tinyml. ACM Computing Surveys **56**, 259–296 (Jun 2024)
38. Wang, Y., Li, M., Cai, H., Chen, W.M., Han, S.: Lite pose: Efficient architecture design for 2d human pose estimation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 13126–13136 (2022)