6

Pareto Optimal Benchmarking of AI Models on ARM Cortex Processors for Sustainable Embedded Systems

Pranay Jain[1,2], Maximilian Kasper[1], Göran Köber[2], Oliver Amft[2,3], Axel Plinge[1], and Dominik Seuss[1,4]

[1]Fraunhofer Institute for Integrated Circuits IIS, Germany
[2]Intelligent Embedded Systems (IES) - Lab, University of Freiburg, Germany
[3]Hahn-Schickard, Germany
[4]Center for Artificial Intelligence and Robotics - Technical University of Würzburg-Schweinfurt, Germany

Abstract

This work presents a practical benchmarking framework for optimizing AI models on ARM Cortex processors (M0+, M4, M7), focusing on energy efficiency, accuracy, and resource utilization in embedded systems. Through the design of an automated test bench, we provide a systematic approach to evaluate across key performance indicators (KPIs) and identify optimal combinations of processor and AI model. The research highlights a near-linear correlation between floating-point operations (FLOPs) and inference time, offering a reliable metric for estimating computational demands. Using Pareto analysis, we demonstrate how to balance trade-offs between energy consumption and model accuracy, ensuring that AI applications meet performance requirements without compromising sustainability. Key findings indicate that the M7 processor is ideal for short inference cycles, while the M4 processor offers better energy efficiency for longer inference tasks. The M0+ processor, while less efficient for complex AI models, remains suitable for simpler tasks. This work provides insights for developers, guiding them

to design energy-efficient AI systems that deliver high performance in real-world applications.

Keywords: Edge AI, energy benchmarking, sustainable, deep compression.

6.1 Introduction

The integration of artificial intelligence (AI) into embedded systems is challenged by the need to balance model performance with energy consumption, a crucial factor for the sustainability and practicality of these systems. Edge AI provides energy efficiency, low latency, and privacy directly on the device, making them critical for applications from smart home devices to autonomous vehicles [7, 8].

As shown in Table 6.1, Edge AI platforms operate under strict resource constraints, with significantly less memory, power, and a smaller CO2 footprint compared to their cloud and mobile counterparts [1]. While considerable research has been conducted on benchmarking AI performance on single-board computers (SBCs) like Raspberry Pi, there is a gap in studies focusing on bare-metal processors. SBCs, while popular and accessible, introduce additional layers of abstraction that can obscure the true performance characteristics of the hardware. In contrast, bare-metal processors provide a more direct and granular understanding of the hardware capabilities, free from the overhead introduced by operating systems and middleware. This study addresses the need for a dedicated test bench that evaluates embedded AI systems at the bare-metal level. Key performance indicators such as accuracy, inference time, and energy consumption are measured, and Pareto front analysis is applied to identify optimal trade-offs. The results provide a solid basis for selecting AI models that achieve high efficiency even under strict resource constraints.

The rest of the paper is structured as follows. Section 2 reviews related work and highlights the need for standardized benchmarking in embedded AI. Section 3 outlines the methodology, including the design of the benchmarking framework, model selection, and experimental setup. Section 4 presents the

Table 6.1 Cloud vs Mobile vs Edge AI systems across different parameters [1].

Platform	Freq.	Memory	Storage	Power		Price	CO2 Footprint
Cloud	GHz	10+ GB	TBs-PBs	~1	kW	1000$+	Hundreds of kgs
Mobile	GHz	Few GB	GBs	~1	W	100$+	Tens of kgs
Edge AI	MHz	KBs	Few MB	~1	mW	10$+	Single kgs

benchmarking results, with a focus on trade-offs between accuracy, latency, and energy efficiency. Finally, Sections 5 and 6 suggest directions for future research and conclude the paper.

6.2 Related Work

This work is positioned at the intersection of sustainability in Edge AI systems and the benchmarking of AI models on embedded platforms. Existing literature provides valuable insights into these domains, yet notable gaps remain that motivate the contributions of this study.

The sustainability of Edge AI has gained attention as the growth of IoT devices contributes to rising carbon emissions and electronic waste [1]. Performing AI inference directly on resource-constrained microcontrollers offers significant reductions in energy use compared to cloud-based computation. The life cycle assessment study by [1] highlights the potential for this approach to lower emissions across diverse applications but also points out its non-trivial footprint when scaled globally. Their findings underline the need for energy-aware design principles that consider the environmental trade-offs of deploying Edge AI at scale. However fine-grained studies on bare-metal processors remain limited.

Benchmarking AI models on embedded systems has been widely studied, with challenges including restricted memory, limited energy budgets, and heterogeneous architectures [2]. MLPerf Tiny [3] provides a standardized suite for assessing the latency, accuracy, and energy efficiency of small-scale models, ensuring reproducibility across platforms. DeepEdgeBench [4] expands this scope by evaluating multiple neural networks across accelerators and processors, focusing on inference time, memory footprint, and energy use. LwHBench [5] further benchmarks lightweight models on single-board computers (SBCs), analyzing inference speed, accuracy, and power consumption to guide deployment decisions. These frameworks have advanced evaluation practices for embedded AI, but they concentrate largely on SBCs and higher-level platforms, where operating systems and middleware can obscure hardware-specific behavior.

Model compression techniques particularly structured pruning [6] and static quantization [7] are widely used to reduce model size and energy consumption for edge deployment. Meanwhile, multi-objective Bayesian optimization has emerged as a powerful tool to navigate trade-offs between accuracy, latency, and energy [8]. However, to the best of the authors knowledge these methods have not been systematically evaluated in bare-metal settings.

6.3 Methodology

The proposed test bench is depicted in Figure 6.1 and follows a structured workflow that systematically evaluates AI models on bare-metal processors to achieve energy-efficient embedded AI design. The workflow consists of four key stages: Model Generation, Model Deployment, System Benchmarking, and Pareto Optimal Solution, each contributing to the comprehensive assessment of AI models in resource-constrained environments.

The workflow begins in the Model Generation stage, starting with a baseline architecture for a given use case. To explore various trade-offs between performance and resource requirements, we employ an automated multi-objective optimization method to guide the application of structured pruning [6, 8]. This process generates a diverse population of pruned models. Following this, each model undergoes static 8-bit quantization to further reduce memory footprint and improve computational efficiency [7]. The final output of this stage is a collection of multiple models in the ONNX format, each representing a unique trade-off point.

The Model Deployment stage converts the ONNX models into executable binaries tailored for the target hardware. First, each ONNX model is translated into a self-contained C model library. This library is then integrated with a C++ benchmarking framework designed to control the inference execution

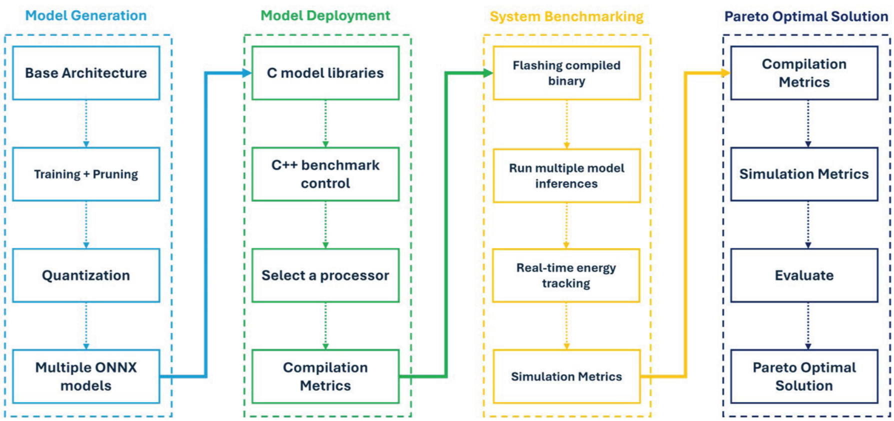

Figure 6.1 Overview of the Test Bench Architecture and Workflow.

Table 6.2 Overview of Benchmark AI Use-Cases.

Use-case	Model	#Params	Dataset	Quality Target
Image Classification	ResNet [3], [9]	**78k**	CIFAR-10 [10]	≥80%
Optical Digit Recognition	LeNet-5 [11]	**140k**	MNIST [11]	≥95%
Anomaly Detection	Autoencoder [3], [12]	**269k**	ToyADMOS [13]	AUC ≥0.85
Visual Wake Words	MobileNetV1 [14]	**3198k**	MSCOCO14 [15]	≥80%

and data collection. The test bench allows for the selection of a specific target processor (Cortex-M0+, M4, or M7), ensuring the code is compiled for the correct architecture. Finally, the C++ control code and the C model are compiled together, producing a single executable binary ready for the System Benchmarking stage.

In the System Benchmarking stage, the compiled binary is flashed onto the target processor. The benchmarking framework then executes multiple model inferences to gather stable performance data. During this execution, we use real-time energy tracking to precisely measure power consumption, while simultaneously recording key inference metrics such as execution latency.

Finally, the Pareto Optimal Solution stage processes the benchmarking results to determine the most efficient AI model configuration by analyzing compilation metrics (e.g. computational overhead) and inference metrics (e.g., latency, accuracy, and power consumption), while also considering use case and hardware specific factors such as idle time between inference cycles, since idle power consumption can significantly affect overall efficiency. Through this systematic evaluation, models are compared to identify optimal trade-offs, ensuring that the final deployed AI model achieves the best balance between energy efficiency and performance for resource-constrained embedded environments.

This structured workflow provides a systematic approach to benchmarking AI models on bare-metal processors, enabling precise evaluation and optimization for energy-efficient Edge AI applications.

6.3.1 Use-Cases

The selected use-cases are representing diverse application domains relevant to embedded AI benchmarking and are adopted from [3]. Optical digit recognition was additionally included and serves as a foundational computer vision task, leveraging LeNet-5 on MNIST [11]. We set a target accuracy of 95%, as MNIST is largely considered a solved task but remains a useful benchmark for assessing high-accuracy performance under edge constraints. Anomaly detection employs an Autoencoder trained on industrial sound datasets [3, 12, 13], targeting robust detection when only normal data is available. Compact image classification relies on a customized ResNet architecture [3, 9] optimized for embedded constraints, evaluated on CIFAR-10 [11]. Finally, Visual Wake Words uses MobileNetV1 [14] for binary person detection on MSCOCO [15], reflecting IoT-oriented scenarios.

6.3.2 Experimental Setup

The experimental setup is designed to systematically evaluate the performance and energy efficiency of AI models deployed on bare-metal processors. As illustrated in Figure 6.2, the setup consists of key hardware components, including an ATP carrier board, a Segger J-Link debugger, a Power Profiler Kit (PPK), and associated signal connections to facilitate accurate measurement of inference-related metrics.

The ATP carrier board serves as the primary hardware platform for executing AI models across different embedded processors. It supports multiple processor configurations, including M0+, M4, and M7 cores, enabling comparative benchmarking of AI models under varying computational capabilities. The deployment process involves flashing the model.hex file onto

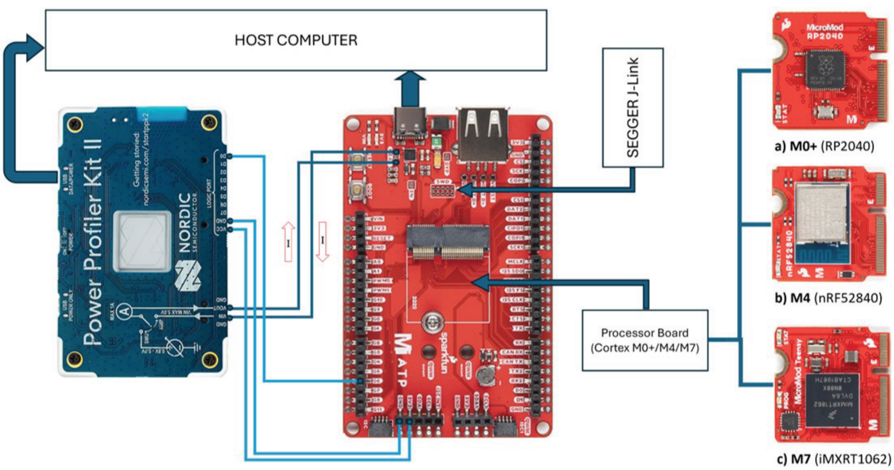

Figure 6.2 Experimental Setup for Test Bench Evaluation.

the respective processor. For the M0+ and M4 processors, the Segger J-Link debugger is used for programming, ensuring precise execution and debugging capabilities. In contrast, the M7 processor is flashed via the USB-C interface, providing a streamlined deployment process.

To measure energy consumption during model inference, a Power Profiler Kit (PPK) is integrated into the setup. The PPK's Vin and Vout pins are connected to the MEAS pins on the ATP carrier board, enabling real-time measurement of the current flowing through the board. This configuration provides critical insights into the power and energy consumption across different AI models and hardware configurations. Additionally, to accurately mark the inference execution phases, a GPIO pin on the ATP carrier board is connected to the D0 pin of the PPK. This connection allows the PPK's state to be toggled as shown in Figure 6.3, ensuring precise synchronization of power measurements with inference execution.

The flow diagram on the left in Figure 6.3 illustrates the control sequence for toggling a GPIO pin and executing multiple inferences per active phase. The plot on the right depicts the measured current consumption (red) and inference state (blue) over time, with the dashed green line indicating the mean current during active processing. Within each cycle, repeated inference execution ensures sufficient measurement duration and enables averaging of current values across multiple runs, resulting in more reliable and representative measurements.

This experimental setup ensures a robust and reliable evaluation of AI model efficiency on embedded processors by facilitating direct power measurements and enabling comparative performance analysis across different core architectures.

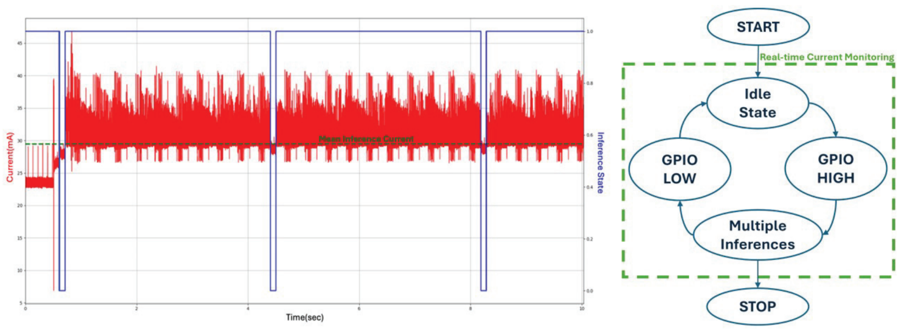

Figure 6.3 Benchmarking flow diagram (left) and exemplary current measurement (right).

6.4 Results

This chapter presents the key results of the study, including test-bench reliability, model variation, the relationship between FLOPs and inference time, and a Pareto analysis of varying inference cycle times to reveal trade-offs between energy consumption and model accuracy.

6.4.1 Test-bench Reliability

To validate the reliability of our test bench, each AI model was benchmarked five times to assess the consistency of the measurements. The variance across key metrics like Inference Current, Time, and especially Energy was minimal. This negligible variability confirms the stability and robustness of our experimental setup, ensuring the trustworthiness of the subsequent findings.

6.4.2 Model Size across use-cases

Figure 6.4 shows the ROM requirements of quantized and unquantized models across the benchmarked use cases (log scale). Quantization yields a substantial reduction in model size, often to one-quarter of the original, making deployment on constrained devices feasible. Among the tasks, Optical Digit Recognition is the smallest, while Visual Wake Words is nearly 50× larger, highlighting the wide range of memory footprints. Notably, unquantized Visual Wake Words models exceeded the available memory on the target processors.

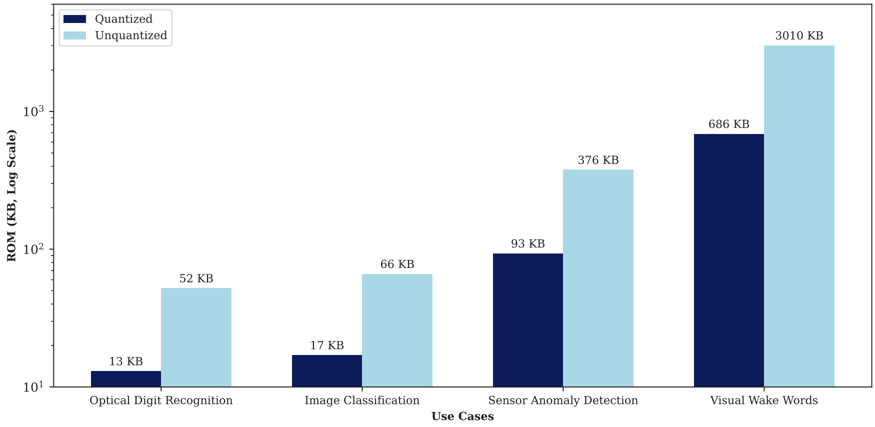

Figure 6.4 Mean AI Model Sizes per use-case.

6.4.3 Correlation Between FLOPS and Inference Time

Through our experiments, RAM and ROM usage were found to be poor predictors of energy consumption and are therefore considered primarily as hardware constraints that determine whether a model can be deployed on a given device. In contrast, Figure 6.5 shows a linear relationship between FLOPS and inference time. This correlation indicates that models with higher FLOPS tend to have longer inference times, highlighting the importance of FLOPS as a critical metric in evaluating model efficiency. Understanding this relationship helps to anticipate the computational demands of the models and make informed decisions about resource allocation and optimization strategies.

The analysis of the relationship between FLOPS and inference time for the M0+, M4, and M7 processors (Figure 6.5) reveals a strong linear correlation ($R^2 \geq 0.93$). This indicates that FLOPS can serve as a reliable predictor of inference time within the evaluated range. The results from Table 6.3 highlight that while all processors exhibit strong linear scaling, their computational efficiencies differ substantially—the M0+ and M4 show a higher sensitivity, whereas the M7 achieves significantly faster inference per FLOP due to its more advanced architecture.

This established relationship supports using FLOPS as a predictive metric for inference time on embedded platforms. As illustrated in Figure 6.6, this

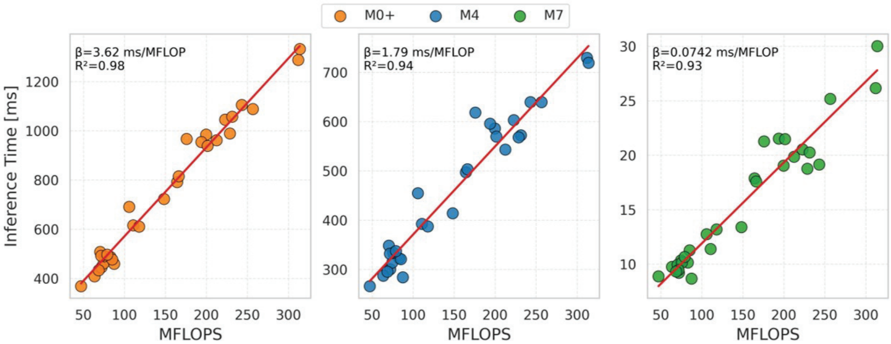

Figure 6.5 Linear Dependency of FLOPS on Inference Time.

Table 6.3 Linear Regression Model Fit between FLOPs and Inference Latency.

Processor	β [ms/MFLOPs]	R^2
M0+	3.620	0.98
M4	1.790	0.94
M7	0.074	0.93

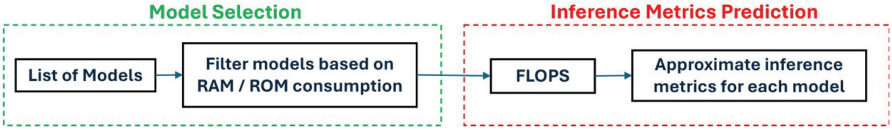

Figure 6.6 Process Flow for Model Selection and Inference Metrics Prediction.

allows developers to estimate performance early in the design phase: once model feasibility is confirmed via RAM and ROM constraints, FLOPS can be used to approximate expected inference latency. This approach facilitates processor–model co-design, enabling more efficient benchmarking and selection in constrained environments.

For instance, if a particular processor's RAM and ROM capacity align with the requirements of a given model, its FLOPS can then be used to approximate the expected inference time and inference energy. This benchmarking analysis not only validates the use of FLOPS as a predictive metric for inference time but also highlights the potential for using it in conjunction with RAM and ROM specifications to guide the selection of the most appropriate processor for specific applications.

6.4.4 Analysis of Inference Cycle energy

Our energy efficiency analysis focuses on the total energy consumed per inference cycle, a metric combining both the active inference period and the subsequent idle time. The energy consumed during the idle state is especially critical in applications with infrequent events, as it can dominate the total power budget.

The processors exhibit significant differences in their datasheet deep sleep currents. The Cortex-M4 is highly optimized for low-power states with an idle current of just 0.30 mA, making it far more efficient during inactivity than the M0+ (4.20 mA) and the M7 (1.60 mA). This idle current profoundly impacts overall efficiency, as demonstrated when analysing the key performance indicators (KPIs) of the full inference cycle.

Figure 6.7 shows the total inference cycle energy for each processor over varying cycle times (0–5 s), comparing the smallest and largest models obtained from our optimization to illustrate the performance range. Because the supply voltage remains constant at 3.3 V, power trends directly reflect current measurements.

The data reveal distinct energy–performance characteristics. The M0+ processor consistently exhibits the highest energy consumption due to its

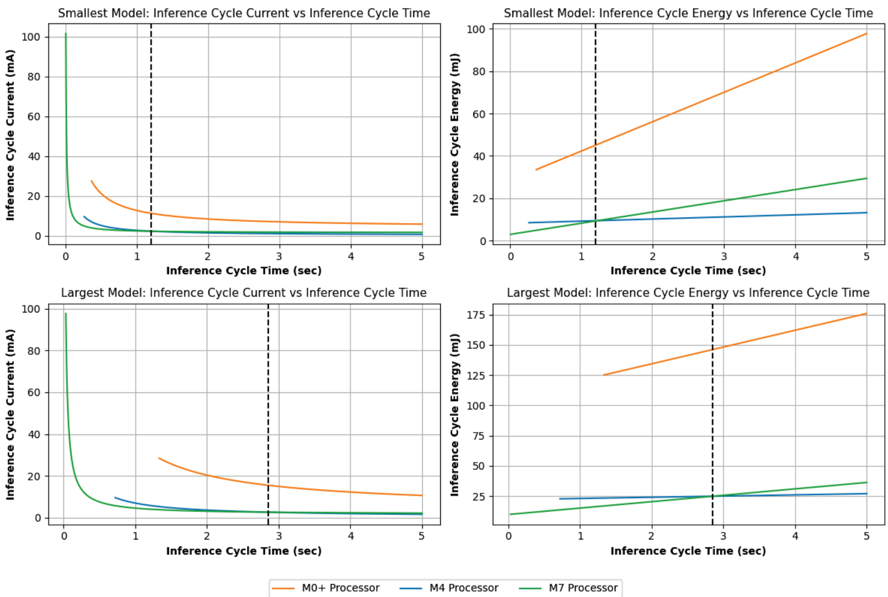

Figure 6.7 Inference Cycle Current and Energy vs Cycle Time across processors for the smallest and largest models.

elevated inference and idle currents. In contrast, the M4 processor becomes increasingly efficient at longer cycle times, where its low idle current dominates the energy budget. The M7processor, benefiting from its high computational speed, achieves the lowest energy consumption during short inference cycles by minimizing active power-on duration.

These results emphasize the trade-offs between processor architectures: the M7 is best suited for short, frequent inference tasks, while the M4 offers superior efficiency for longer, intermittent workloads. The M0+ remains the least efficient across all scenarios.

6.4.5 Analysis of Energy Efficiency and Accuracy Trade-offs

Given its consistently higher inference and idle currents, the M0+ processor was excluded from further energy–accuracy analysis, as it offers no competitive advantage under the evaluated constraints. Figure 6.8 presents a Pareto front analysis comparing inference cycle energy and accuracy across varying cycle times (0.5 s, 2.5 s, 5.0 s) for the M4 and M7 processors.

The results reveal a fundamental shift in design priorities depending on how frequently inference occurs. When inference tasks are frequent—such as

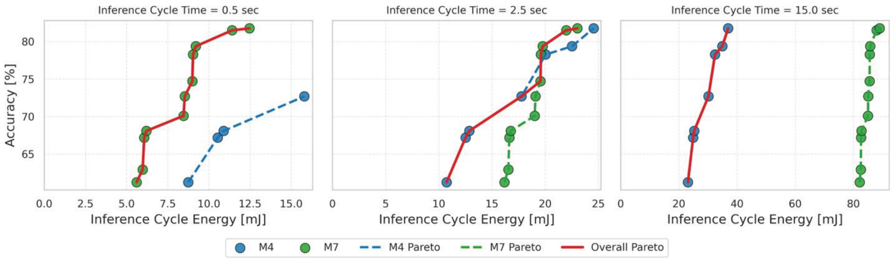

Figure 6.8 Energy vs. accuracy trade-offs for M4 and M7 across short, medium, and long inference cycles. Pareto fronts highlight optimal choices per workload.

in applications with short cycle times (e.g., ≤0.5 s)—the energy consumed during active computation dominates the total budget. In this regime, the Cortex-M7's high computational throughput allows it to complete inference significantly faster, resulting in lower total energy for a given level of accuracy. Here, model efficiency directly translates into system-level savings, and even modest reductions in inference time can yield meaningful energy benefits. Consequently, models should be optimized primarily for speed, with accuracy maintained only to the extent necessary for functional correctness.

In stark contrast, when inference is infrequent and long idle periods separate executions—as in applications with cycle times of 2.5 s or more—the energy consumed while waiting far outweighs that used during computation. Under these conditions, the Cortex-M4's ultra-low idle current becomes decisive, making it the more energy-efficient choice despite its slower inference speed. More importantly, in this idle-dominated regime, the marginal energy cost of running a slightly slower but more accurate model becomes negligible. The execution time contributes little to the total energy budget, so there is little to gain from aggressive model compression or latency optimization. Instead, designers should favour models that maximize accuracy, as the performance benefit during the rare inference event outweighs the minimal energy penalty of a longer active phase.

This duality underscores a critical principle for embedded AI deployment: the optimal balance between energy and accuracy is not fixed—it is dictated by the application's temporal behaviour. Processor and model selection must therefore be guided not only by hardware capabilities and model metrics, but also by the expected duty cycle of the target use case. The Pareto fronts in Figure 6.8 thus serve not just as performance benchmarks, but as a decision map for co-designing hardware, models, and application timing.

6.5 Discussion and Future Work

This study provides embedded AI developers with a practical, application-driven framework for selecting hardware and optimizing models — moving beyond simplistic "faster is better" assumptions. Instead, the optimal choice depends on how a processor's energy profile aligns with the temporal structure of the target application.

Inference cycle duration emerges as the primary determinant of architectural preference. For applications demanding frequent, low-latency inference, such as real-time industrial monitoring ($\leq$0.5 s cycles), the Cortex-M7's high computational throughput dominates energy efficiency. Its DSP-enhanced architecture minimizes active execution time, making it ideal when inference is the dominant energy sink. Conversely, for battery-powered devices with long idle intervals, such as environmental sensors ($\geq$5 s cycles), the Cortex-M4's ultra-low idle current becomes decisive. Here, minimizing power during waiting periods outweighs the benefit of faster computation. The Cortex-M0+, while unsuitable for frequent-inference AI workloads due to its high idle consumption, retains value in non-AI contexts and DIY projects, where cost, accessibility, and development simplicity are prioritized over performance.

Critically, this dichotomy informs not only processor selection, but also model optimization strategy. In short-cycle regimes, where active computation dominates energy use, reducing inference latency directly improves system efficiency favouring compact, fast models. In long-cycle regimes, however, the energy cost of inference becomes negligible compared to idle consumption. Here, developers can prioritize accuracy over latency, selecting higher-performing models without significant energy penalty.

To aid this complex decision-making process, our work establishes two practical tools. First, FLOPs serve as a reliable early-stage predictor of inference latency, allowing for hardware-aware model selection without exhaustive testing. Second, the Pareto front, which visually maps the energy–accuracy trade-offs per cycle time, guiding developers toward optimal processor-model pairings.

6.5.1 Limitations

Despite its practical insights, this study has several limitations. Power measurements were conducted under controlled laboratory conditions and may not fully capture variability introduced by environmental factors, peripheral usage, or long-term deployment effects. In addition, the analysis focuses on a limited set of microcontroller architectures, which may restrict the generalizability of the conclusions to newer or heterogeneous platforms. Finally, while FLOPs correlate well with inference latency in the evaluated setups, this relationship may weaken for models with complex memory-access patterns or hardware-specific accelerations.

6.5.2 Future Work

Future research should focus on refining power-measurement methodologies, particularly through standardized evaluations of idle current and wake-up latency to enable fair cross-processor comparisons. Furthermore, long-term, real-world deployments are required to validate the robustness of these findings under environmental variation, workload drift, and hardware aging. Extending the framework to additional architectures and accelerator-based systems would further strengthen its applicability.

6.6 Conclusion

This work presented a framework for benchmarking AI models on bare-metal ARM Cortex processors, revealing that the optimal system design is dictated by the application's operational cycle. The core finding is that the balance between active computation energy and passive idle energy not only determines the best processor choice but also shapes the ideal model optimization strategy.

For frequent-inference tasks, the Cortex-M7 is superior, and model efficiency is critical for minimizing energy use. For tasks with long idle periods, the Cortex-M4 is the clear winner, allowing developers to prioritize model accuracy over inference speed with minimal energy penalty. Our framework, which uses FLOPs as a latency predictor and Pareto analysis to visualize trade-offs, provides a structured basis for navigating these decisions. Ultimately, these findings guide developers toward creating embedded AI applications that are both sustainable and high performing by aligning hardware selection and model optimization with the specific demands of the real-world use case.

Acknowledgement

This work is part of the GreenICT@FMD project and is funded by the German Federal Ministry for Research, Technology and Space (BMFTR) (grant number 16ME0491K).

References

[1] S. Prakash *et al.*, "Is TinyML Sustainable? Assessing the Environmental Impacts of Machine Learning on Microcontrollers," *arXiv.org*, 2023. https://arxiv.org/abs/2301.11899

[2] C. R. Banbury *et al.*, "Benchmarking TinyML Systems: Challenges and Direction," *arXiv:2003.04821 [cs]*, Jan. 2021, Available: https://arxiv.org/abs/2003.04821.

[3] C. R. Banbury *et al.*, "MLPerf Tiny Benchmark," *arXiv (Cornell University)*, Jun. 2021, doi: https://doi.org/10.48550/arxiv.2106.07597.

[4] S. P. Baller, A. Jindal, M. Chadha, and M. Gerndt, "DeepEdgeBench: Benchmarking Deep Neural Networks on Edge Devices," *arXiv.org*, 2021. https://arxiv.org/abs/2108.09457.

[5] A. Garcia-Perez, R. Miñón, A. I. Torre-Bastida, and E. Zulueta-Guerrero, "Analysing Edge Computing Devices for the Deployment of Embedded AI," *Sensors (Basel, Switzerland)*, vol. 23, no. 23, p. 9495, Nov. 2023, doi: https://doi.rog/10.3390/s23239495.

[6] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," *arXiv:1510.00149 [cs]*, Feb. 2016, Available: https://arxiv.org/abs/1510.00149.

[7] B. Jacob *et al.*, "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, doi: https://doi.org/10.1109/cvpr.2018.00286.

[8] M. Deutel, G. Kontes, C. Mutschler, and J. Teich, "Combining Multi-Objective Bayesian Optimization with Reinforcement Learning for TinyML," *ACM Transactions on Evolutionary Learning and Optimization*, Jan. 2025, doi: https://doi.org/10.1145/3715012.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE conference on computer vision and pattern recognition*, Las Vegas, NV, USA, 2016, pp. 770–778, doi: https://doi.org/10.1109/CVPR.2016.90.

[10] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10," *University of Toronto, Tech. Rep.,* 2009, https://www.cs.toronto.edu/~kriz/cifar.html.

[11] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 2018, doi: https://doi.org/10.1109/5.726791.

[12] G. Li and J. J. Jung, "Deep learning for anomaly detection in multivariate time series: Approaches, applications, and challenges," *Information Fusion*, vol. 91, pp. 93–102, Mar. 2023, doi: https://doi.org/10.1016/j.inffus.2022.10.008.

[13] Y. Koizumi, S. Saito, H. Uematsu, N. Harada, and K. Imoto, "ToyADMOS: A dataset of miniature-machine operating sounds for anomalous sound detection," *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, NY, USA, 2019, pp. 313–317, doi: https://doi.org/10.1109/WASPAA.2019.8937164.

[14] Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv (Cornell University)*, Apr. 2017, doi: https://doi.org/10.48550/arxiv.1704.04861.

[15] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," *Computer Vision – ECCV 2014*, vol. 8693, pp. 740–755, 2014, doi: https://doi.org/10.1007/978-3-319-10602-1_48.