7

Improving Classifier Latency at the Edge through ARM Helium

Lorenzo Abate, Mario Barbareschi, and Antonio Emmanuele

Università degli Studi di Napoli Federico II, Italy

Abstract

The increasing diffusion of intelligent devices at the network edge has led to a growing demand for efficient on-device inference, capable of overcoming the limitations of traditional cloud-centric computing paradigms. This work investigates the acceleration of **decision tree–based inference** on **resource-constrained edge platforms** by exploiting **ARM Helium vector extensions**, which bring the **Single Instruction Multiple Data (SIMD)** paradigm to the **Cortex-M** class of processors.

A dedicated **SIMD-based kernel** was implemented and tested on the **NUCLEO-STM32N657** board across three UCI datasets (AI4I, Dry Bean, Avila). Results show up to **$\sim$15% latency reduction** over the **non-SIMD baseline**, confirming that **ARM Helium** effectively exploits data-level parallelism to enhance inference efficiency on lightweight microcontrollers.

Overall, this study provides experimental evidence that **vector extensions represent a key enabler** for bringing advanced machine learning capabilities to low-power embedded systems, bridging the gap between traditional microcontroller efficiency and modern AI acceleration at the edge.

Keywords: Edge AI, SIMD, Classification, Random Forest.

7.1 Introduction

A **cloud-centric computing paradigm** has traditionally been adopted for Edge devices, where data collected by end nodes are transmitted to large-scale

data centers for processing [1]. This model has largely been driven by the intrinsic constraints of those devices, which are typically characterized by limited memory resources, stringent power-consumption requirements, and modest computational capabilities [2, 3, 4]. However, the exponential growth in the number of connected devices [5, 6]—and the resulting increase in network bandwidth demand—combined with modern requirements for low-latency processing, has rendered an alternative paradigm increasingly appealing: **executing inference directly on the device** [7].

One of the most widely adopted model families at the edge is that based on **binary decision trees**, such as **Random Forests**. These models are highly valued for their ability to handle both **classification** and **regression** tasks while remaining **lightweight**, **interpretable**, and suitable for deployment on resource-constrained platforms [8].

In the literature, several approaches have been proposed to enhance the performance of tree ensembles on edge devices. However, recent **advancements in semiconductor manufacturing** and **processor design** have introduced new opportunities, notably through the integration of **vector extensions**. These technologies make it possible to further improve execution efficiency by exploiting **data-level parallelism**, thereby pushing the boundaries of machine learning performance on embedded and edge platforms.

In particular, this work focuses on the **newly introduced ARM Helium vector extensions**, specifically designed for **edge computing**. Through a **latency analysis** of the inference process, it will be shown how these technologies can **further enhance performance**, demonstrating their potential to significantly accelerate machine learning workloads on resource-constrained devices.

7.2 Vectorial Extensions and ARM Helium

The electronics industry is continuously driving technological advancements, enabling the integration of increasingly sophisticated features even at the edge, such as **vector extensions** [9]. This technology implements the **Single Instruction Multiple Data (SIMD)** paradigm, allowing a single instruction to be executed in parallel on multiple data elements, thereby effectively enabling **data-level parallelism**. Vector extensions **extend the standard processor instruction set** with specialized vector registers and operations capable of processing multiple operands simultaneously. Instead of performing the same arithmetic or logical operation repeatedly on individual data points, the processor can apply it once to an entire vector of values.

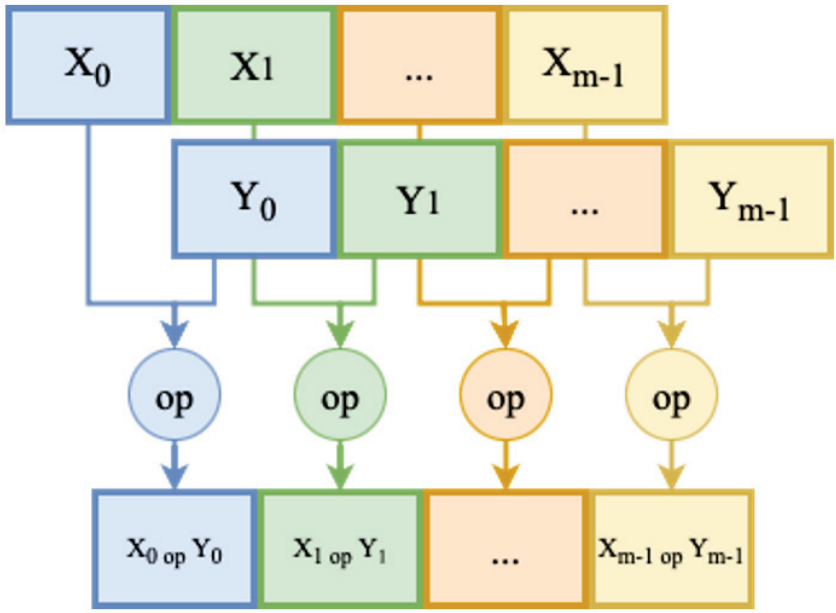

Figure 7.1 SIMD Execution

This approach significantly **improves throughput** in workloads characterized by regular, repetitive computations — such as signal processing, image analysis, and machine learning inference — by exploiting the inherent parallelism in data streams while maintaining a compact code structure and reducing execution time.

Figure 7.1 illustrates SIMD execution that leverages wide vector registers that are divided into multiple **lanes**. Each lane holds one data element, and the same instruction is applied simultaneously across all lanes, producing multiple results in parallel. The number of lanes depends on how the register width is partitioned according to the data type.

Over the years, numerous types of **vector extensions** have been introduced [9], each characterized by distinct design choices and architectural trade-offs. Among these, the **ARM Helium extensions** [10] are particularly relevant for this work, as they bring the **SIMD paradigm** to the **Cortex-M** family of processors, thereby enabling efficient parallel execution on resource-constrained **edge devices**.

Helium features **eight 128-bit vector registers**, which can be partitioned into a variable number of **lanes** depending on the data type being processed,

supporting element sizes up to **32 bits** (8/16/32-bit int, 16/32-bit floating point). This flexible register organization allows developers to efficiently exploit the available parallelism according to the precision required by the application.

In addition to providing a rich set of **arithmetic and logical operations** extended over multiple data elements—such as addition, multiplication, and bitwise manipulation—Helium also introduces a comprehensive suite of **data handling and control instructions**. Among these, the **Gather** and **Scatter** operations are particularly noteworthy, as they enable non-contiguous memory access patterns by allowing data to be loaded from or stored to arbitrary memory locations. This capability greatly enhances performance in workloads characterized by **irregular data structures**.

It is precisely within this class of workloads that the present work is positioned, providing an analysis of **latency reduction** in the **inference of binary decision tree–based models** by exploiting the **ARM Helium vector extensions**. The objective is to demonstrate how such extensions can be effectively leveraged to achieve **significant latency improvements**, thereby enhancing the efficiency of machine learning inference on **edge devices**.

7.3 Experimental Results

7.3.1 Experimental Setup

Inference on a **decision tree** can be regarded as a **tree traversal**, where, starting from the **root node**, the computation proceeds toward a **leaf node**, at which the final prediction is produced. The specific path followed during traversal is determined by the outcome of a **comparison** between an **input feature** and a **threshold value** stored at each node [3].

Consequently, the form of **parallelization** that can be exploited lies in the ability to process **multiple nodes simultaneously**, thereby performing several comparisons within a single instruction.

In particular, this section presents a **latency comparison** between two approaches: a **SIMD-based method**, implemented using **ARM Helium**, where multiple nodes are traversed in parallel through a technique inspired by the work of **Kim et al.** [11], and a **Classic tree visiting** approach, which processes one node at a time.

The **SIMD technique** is based on reorganizing the nodes of the tree into **hierarchical blocks**, which can be visualized as **triangular sub-structures** grouping together a fixed number of nodes. The purpose of these blocks is

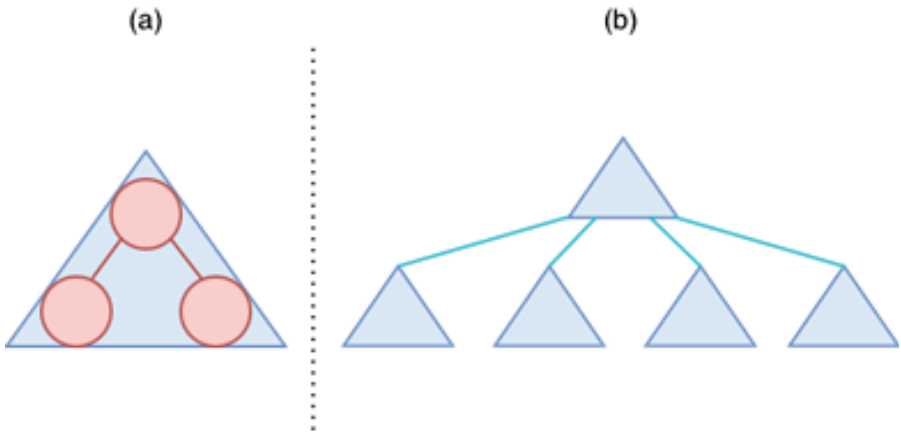

Figure 7.2 (a) example of a block with 4 lanes. (b) example of the resultant tree structure.

to align the tree layout with the **SIMD parallelism level**. For instance, as illustrated in **Figure 7.2 (a)**, a SIMD unit with **four lanes** can accommodate a **block of three** nodes, enabling three comparisons to be executed in parallel. **Figure 7.2 (b)** then shows the structure of the resulting tree.

The experiments were conducted on the **NUCLEO-STM32N657** development board, featuring an **ARM Cortex-M55** processor equipped with **ARM Helium vector extensions**. The core was configured to operate at a frequency of **600 MHz**.

The evaluation employed **Random Forest models** trained on three datasets from the **UCI Machine Learning Repository—AI4I** [12], **Dry Bean** [13], and **Avila** [14]—selected to provide a **heterogeneous benchmarking suite** encompassing diverse data characteristics and complexity levels.

The following table summarizes the main characteristics of the three datasets.

Table 7.1 Summary of the datasets used in the experiments.

Dataset	Samples	Features	Classes
AI4I	1000	6	2
Dry Bean	13610	16	7
Avila	20866	10	12

A **quantization process** was also applied to reduce the model size and make it more suitable for an **edge deployment scenario**. In particular,

quantization-aware training (QAT) [3] was adopted, in which quantization is applied to the input data already during the training phase. As a result, the trained model becomes inherently adapted to the **reduced-precision representation**, maintaining high predictive accuracy while significantly lowering memory and computational requirements. In this work, the data were represented as **int16**.

In particular, the process relies on the computation of a **scaling factor** defined as

$$scale = \frac{2^{bits-1}}{|\max\_val|}$$

where **bits** represent the target bit-width of the new representation and **max_val** is the maximum absolute value within the considered set. Each element is then multiplied by this scaling factor and rounded to the nearest integer, obtaining the new quantized representation. This process is applied **separately** to each feature in the dataset.

The following table provides detailed information on the **models** used for the analysis.

Table 7.2 Summary of the models used in the experiments.

Dataset	Depth	Trees	Nodes	Accuracy
AI4I	10	100	23320	0.985
Dry Bean	10	100	50804	0.922
Avila	10	100	61768	0.878

7.3.2 Timing Analysis

At this stage, the **results of the analysis** are presented. The evaluation was carried out on **1,000 test samples**. The SIMD-based approach was evaluated using a **parallelism level of seven nodes**. With **16-bit** quantized data, each block can accommodate **seven nodes**, as **eight SIMD lanes are available**.

Figure 7.3 illustrates a comparison between the performance of the **SIMD** and **non-SIMD** approaches across the three models introduced in **Table 7.2**. The performance metric is expressed in terms of the **number of clock cycles** required to complete the inference, with the **median value** over the **1,000 test samples** shown as the height of each bar.

The results clearly highlight that the use of **vector extensions** enables a **significant reduction in inference latency**, demonstrating the effectiveness of SIMD parallelism for accelerating decision tree–based workloads on edge devices.

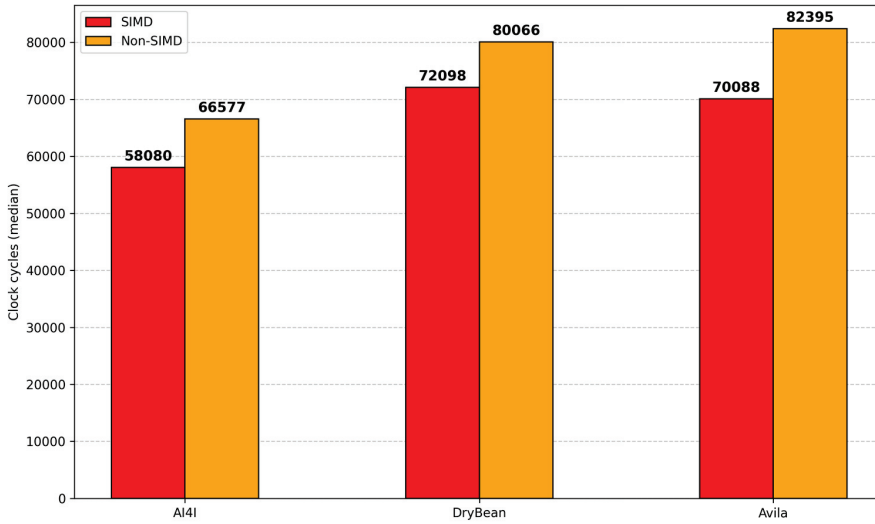

Figure 7.3 Barplot showing the summary of the performance for each model.

Specifically, the results show a **latency reduction** of approximately:

- **13.8%** for the *AI4I* dataset;
- **10%** for the *Dry Bean* dataset;
- **15%** for the *Avila* dataset.

7.4 Conclusion

In conclusion, the results obtained in this study demonstrate the tangible benefits of leveraging **ARM Helium vector extensions** to accelerate **decision tree–based inference** on **edge devices**. By exploiting data-level parallelism, SIMD-based approach achieves a consistent **reduction in latency** across heterogeneous models and datasets, confirming its effectiveness in enhancing computational performance under constrained hardware conditions.

These findings highlight the growing potential of **modern embedded architectures** to support increasingly demanding **machine learning workloads** directly on the device, without relying on cloud offloading. In this context, vector extensions such as **ARM Helium** represent a fundamental step toward **bridging the gap** between high-performance computing and low-power embedded systems, paving the way for more **efficient, autonomous, and intelligent edge computing solutions**.

In future work, we aim to further investigate optimizations based on alternative memory layouts, with the goal of better exploiting the parallelism offered by the **ARM Helium** vector extensions. We also plan to explore deployment strategies that account for the hierarchical memory architectures typical of edge devices.

References

[1] Mario Barbareschi, Antonio Emmanuele, Nicola Mazzocca and F. R. di Torrepadula, "Designing on-board explainable passenger flow prediction," *Engineering Applications of Artificial Intelligence*, vol. 139, p. Art. 109648, 2025. https://doi.org/10.1016/j.engappai.2024.109648.

[2] E. Tabanelli, G. Tagliavini, and L. Benini, "Optimizing random forest-based inference on risc-v MCUs at the extreme edge," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, pp. 4516–4526, Nov. 2022. https://doi.org/10.1109/TCAD.2022.3199903

[3] F. Daghero, A. Burrello, E. Macii, P. Montuschi, M. Poncino, and D. J. Pagliari, "Dynamic decision tree ensembles for energy-efficient inference on iot edge nodes," *IEEE Internet of Things Journal*, vol. 11, no. 1, pp. 742–757, 2024. https://doi.org/10.1109/JIOT.2023.3286276

[4] M. Barbareschi, A. Emmanuele, "A margin based early-stopping approach for random forest classifiers," in: 2025 55th *Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pp. 123–126, IEEE, 2025. https://doi.org/10.1109/DSN-W65791.2025.00050

[5] M. Barbareschi, S. Barone, A. Emmanuele, and N. Mazzocca, "Exploiting Functional Approximation on Decision-Tree Based Multiple Classifier Systems," *in 2024 IFIP/IEEE 32nd International Conference on Very Large Scale Integration (VLSI-SoC),* Oct. 2024, pp. 1–4, iSSN: 2324-8440. https://ieeexplore.ieee.org/document/10767841/?arnumber=10767841

[6] I. Cvitić, D. Peraković, M. Periša, M. Krstić, and B. Gupta, "Analysis of IoT Concept Applications: Smart Home Perspective," in *Future Access Enablers for Ubiquitous and Intelligent Infrastructures*, D. Perakovic and L. Knapcikova, Eds., Cham: Springer International Publishing, 2021, pp. 167–180. https://doi.org/10.1007/978-3-030-78459-1_12

[7] W. Shi, J.Cao,Q. Zhang,Y. Li, andL.Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal,* vol. 3, pp.637–646, Oct. 2016. https://doi.org/10.1109/JIOT.2016.2579198

[8] E.Tabanelli, G.Tagliavini, and L.Benini, "Dnn is not all you need: Parallelizing non-neural ML algorithms on ultra-low-power IoT processors," *arXiv preprint arXiv:2107.09448*, 2021. https://arxiv.org/abs/2107.09448

[9] H. Amiri and A. Shahbahrami, "SIMD programming using intel vector extensions," *Journal of Parallel and Distributed Computing*, vol.135, pp.83–100, 2020. https://www.sciencedirect.com/science/article/pii/S074373151830813X

[10] J.Marsh, *Arm® Helium™ Technology: M-Profile Vector Extension (MVE) for Arm® Cortex®-MProcessors*. Cambridge, UK: Arm Education Media, 2020. https://www.arm.com/resources/education/books/mve-reference-book

[11] C.Kim, J.Chhugani, N. Satish, et al., "FAST: Fast architecture sensitive tree search on modern CPUs and GPUs," *in Proceedings of the 2010 ACM SIGMOD International Conference on Manage ment of Data (SIGMOD'10)*, NewYork, NY, USA: Association for Computing Machinery, June 2010, pp. 339–350. https://doi.org/10.1145/1807167.1807206

[12] UCI Machine Learning Repository, "Ai4i 2020 predictive maintenance dataset [dataset]," 2020. https://doi.org/10.24432/C5HS5C

[13] UCI Machine Learning Repository, "Drybean [dataset]," 2020. https://doi.org/10.24432/C50S4B

[14] C. Stefano, F. Fontanella, M. Maniaci, and A. Freca, "Avila [dataset]," 2018. https://doi.org/10.24432/C5K02X