TinyML for On-Board Earth Observation: From Ultra-Low-Power MCUs to In-Sensor Computing

Luigi Capogrosso*, Pietro Bonazzi†, Loris Hoxhaj‡, Michele Magno†*
*Interdisciplinary Transformation University of Austria, †ETH Zurich, ‡University of Verona

***Abstract*—The rapid democratization of access to Low Earth Orbit (LEO) has driven an exponential increase in Earth Observation (EO) missions powered by miniaturized CubeSats. However, the traditional paradigm of continuously transmitting raw or minimally processed images to ground stations is severely constrained by the stringent power, memory, and communication bandwidth limitations of these tiny systems. To overcome these limitations, we present our recent advances in shifting computational intelligence directly to the extreme edge of space operations for autonomous image classification. The first is a hardware-aware TinyML pipeline for heterogeneous Microcontroller Units (MCUs) featuring an integrated Neural Processing Unit (NPU), which demonstrates massive model compression without compromising accuracy while achieving millijoule-scale energy consumption. The second is an advanced in-sensor computing strategy that executes Deep Neural Networks (DNNs) directly on the image sensor die, completely bypassing the system bus and offloading the primary On-Board Computer (OBC). This article demonstrates how both paradigms can effectively reduce downlink bandwidth requirements and shift task execution to the edge while maintaining acceptable accuracy, paving the way for sustainable, real-time autonomous space robotics.**

I. Introduction

The "New Space" era has democratized access to orbit [1], positioning miniaturized CubeSats as the backbone of numerous Earth Observation (EO) missions [2]. These missions span a wide range of critical applications, including precision agriculture, maritime surveillance, and rapid response to natural disasters [3]. However, CubeSats operate under severe physical constraints in terms of size, weight, power consumption, and communication bandwidth [2]. These inherent limitations fundamentally challenge the traditional operational model of EO, which relies on acquiring massive volumes of Level-0 sensor data and downlinking it to ground stations for centralized processing [4]. Because a typical Low Earth Orbit (LEO) pass lasts only a few minutes, the sheer volume of high-resolution geospatial data vastly exceeds the daily transmission budgets of even the most advanced S-band or X-band radios [5]. Consequently, for these systems, a paradigm shift is required: instead of transmitting raw data to the ground, intelligence must be brought directly to the data.

Empowering satellites to autonomously filter, classify, and prioritize imagery directly in orbit drastically reduces downlink volume and latency for time-critical applications [6], [7]. Although prior efforts have successfully migrated inference tasks to satellite's primary On-Board Computer (OBC) or secondary hardware accelerators, these solutions often rely on relatively powerful Systems on Chips (SoCs) or Field Programmable Gate Arrays (FPGAs) [8], [9]. However, such hardware greatly exceeds the resource envelope available on typical ultra-constrained CubeSats, which often feature only a few megabytes of volatile memory and power budgets of just a few watts [10]. Furthermore, executing models on the primary processing unit still requires the energy-intensive transfer of raw images from the sensor to the internal data bus of the satellite, thereby incurring power penalties and latency overhead [11].

To address these critical gaps, this paper summarizes our recent efforts in developing TinyML pipelines tailored for EO missions. Specifically, the first contribution investigates the optimization of Convolutional Neural Networks (ConvNets) for deployment on an ultra-low-power Microcontroller Unit (MCU) equipped with a dedicated Neural Processing Unit (NPU), which demonstrates massive model compression while maintaining acceptable accuracy and achieving millijoule-scale energy consumption. The second approach pushes the computational boundary even further to the absolute edge by exploiting in-sensor computing, where intelligence is embedded directly into the vision sensor itself. By comparing these methodologies, we demonstrate how TinyML can be a viable solution to alleviate data transmission bottlenecks and overcome the stringent power and processing constraints that currently limit autonomous space robotics.

II. Methodology

A. NPU-based TinyML

Hardware architecture. Experiments were conducted using an STM32N6 [12]. This board is a novel MCU platform that features an Arm Cortex-M55 core at 800 MHz, 4.2 MB of SRAM, and an integrated Neural-ART NPU capable of delivering up to $\sim$600 GOPS for INT8 inference, enhancing the processing capabilities crucial for real-time inference tasks. The use of STMicroelectronics devices in this context constitutes a practical and widely accepted approach in the literature [13]–[16].

We restricted our analysis to ConvNet or NAS-generated ConvNet models for two reasons. First, the ST Edge AI toolchain currently provides optimized operator mapping primarily for these architectures, limiting its compatibility with Transformer-based designs [17]. Second, by focusing on ConvNets, we maximize operator coverage in the Neural-ART NPU [18]. The optimization goals that guide our implementation are the following:

1) Limit the accuracy drop with respect to the Float32 baseline to a task-acceptable margin.
2) Fit the model parameters, intermediate activations, and working buffers within the 4.2 MB SRAM.
3) Meet the per-frame inference deadline defined by the payload control loop on the STM32N6.
4) Minimize energy consumption to extend mission life and increase flexibility in duty-cycling.

Structured iterative pruning. This step alternates between removing filters in convolutional layers or neurons in fully-connected layers, and fine-tuning the network based on the magnitude of their associated weights, which we quantify using the per-layer L_2 Norm [19].

INT8 static quantization. Following the pruning stage, we applied Post-Training Quantization (PTQ) to further compress the model and improve the inference efficiency [20]. This step converts the 32-bit floating-point weights and activations into 8-bit signed integers (INT8).

Hardware-aware operator mapping. Finally, we used the ST Edge AI Developer Cloud framework to perform hardware-aware operator mapping [17], *i.e.*, partitioning the supported Deep Neural Network (DNN) operators into the Neural-ART NPU. The mapping is performed at the operator level to maximize hardware acceleration and minimize latency. Specifically, the tool inspects the quantized ONNX graph [21] and partitions it into subgraphs based on hardware capabilities. Operators such as `Conv2D`, `DepthwiseConv2D`, and `Activations` are transferred to the Neural-ART NPU. Other layers, such as custom or control-flow operations, remain on the CPU [18]. To minimize latency and memory footprint, the optimizer applies:

- *Pipeline Scheduling:* Overlaps CPU and Neural-ART NPU execution where possible.
- *Buffer Reuse:* Shares activation buffers across layers to reduce SRAM usage.
- *Operator Fusion:* Combines compatible layers (*e.g.*, `Conv` + `ReLU`) into single Neural-ART NPU kernel.

This hardware-aware step ensures that the model fully exploits the heterogeneous compute resources of the STM32N6 platform.

B. In-Sensor-based TinyML

Hardware architecture. Experiments were conducted using an IMX500 [22]. This is a novel Sony CMOS sensor that combines image processing and Artificial Intelligence (AI) capabilities on a single chip, reducing infrastructure costs, improving efficiency, and accelerating Edge AI deployment.

INT8 static quantization. To successfully deploy deep learning models onto the IMX500 sensor, the architecture must be processed through a fully hardware-compliant pipeline. Therefore, we utilized the Model Compression Toolkit (MCT) [23], the open-source framework released by Sony Semiconductor Solutions, specifically designed to optimize DNNs on this target platform. This toolkit provides advanced quantization, graph tracing, and compression utilities to bridge the gap between standard deep learning frameworks and in-sensor deployment. Among the optimization techniques supported by the toolkit, we again implemented PTQ. Here, during the PTQ flow, the MCT's quantization core applies multiple algorithmic optimizations, such as operator fusion, optimal step-size calculation, and activation scaling, specifically tailored to the target IMX500 v1.0 hardware specifications [23].

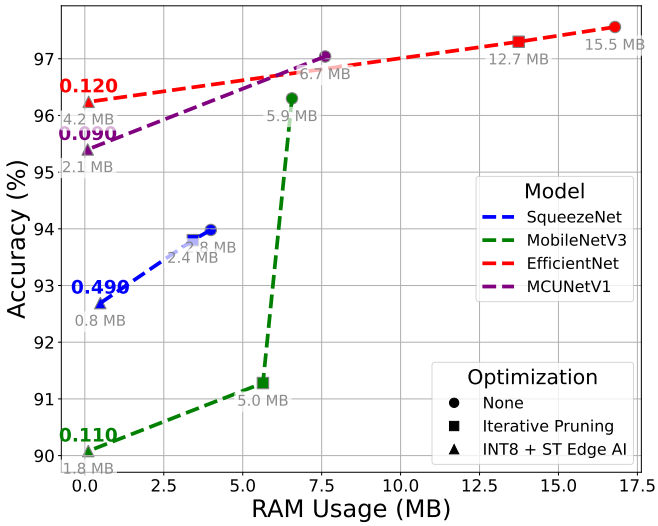

Figure 1: Quantitative results in terms of accuracy, RAM, and Flash (the grey numerical value reported below each data point) used on the EuroSAT dataset.

ONNX graph optimization. Finally, we implemented an additional optimization step to address the architectural incompatibilities between the selected models and the IMX500. Specifically, we manually adjusted the layer attributes in the ONNX graphs to meet the platform's execution requirements, ensuring that all models remained compliant with the target hardware's inference engine.

III. Experiments

A. NPU-based TinyML Results

EuroSAT dataset. The initial evaluation was conducted on the EuroSAT dataset (64×64 images) [24], and the results are presented in Figure 1.

RS_C11 and MEDIC datasets. We extended the experiments to two additional datasets: RS_C11 (128×128) [25] and MEDIC (224×224) [26]. These datasets introduce greater complexity than EuroSAT, both in image resolution and semantic variability. Figure 2 summarizes the results on RS_C11 and MEDIC, respectively.

Discussion. The optimized models maintain a task-acceptable accuracy margin compared to their Float32 baselines, with most accuracy drops remaining below 10 percentage points, thus satisfying the design goal #1. At the same time, memory footprint reductions in RAM (and in Flash) ensure that all model parameters, activations, and buffers fit within the 4.2 MB embedded RAM of the STM32N6, fulfilling the design goal #2.

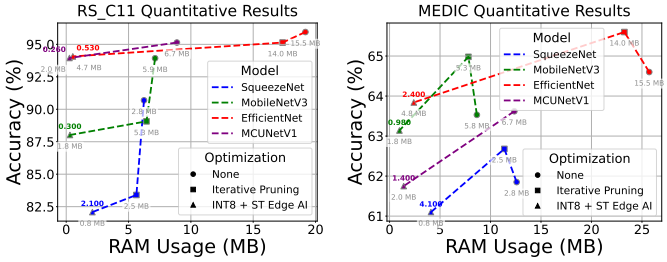

Figure 2: Quantitative results in terms of accuracy, RAM, and Flash (the grey numerical value reported below each data point) used on the RS_C11 (left) and MEDIC (right) datasetS. (*) Zoom in to clearly view the Flash values.

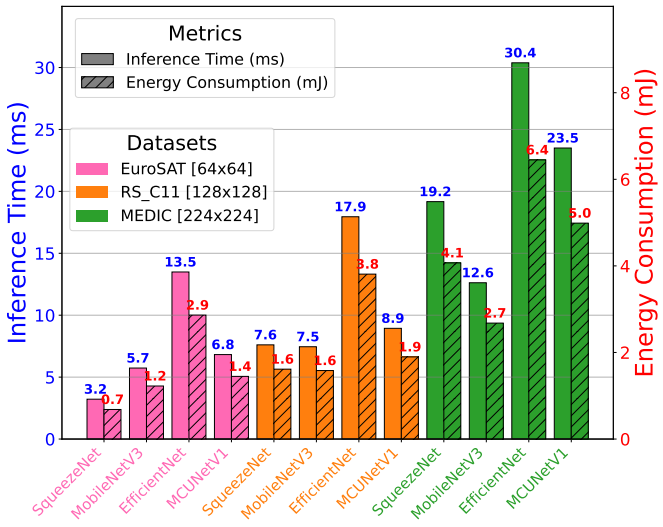

Figure 3: Inference latency and energy consumption for fully optimized models across all the datasets. Left: Latency in milliseconds; Right: Energy per inference in millijoules.

Inference and Power Consumption Analysis. We measured inference time and energy consumption for fully quantized models, with results summarized in Figure 3.

Discussion. These results confirm compliance with the design goals #3 and #4. All optimized models operate within millijoule energy budgets, with the most efficient configuration (SqueezeNet on EuroSAT) consuming only 0.68 millijoules (mJ) per inference. Even in the most demanding scenario (EfficientNet on MEDIC), the energy cost remains within feasible limits for CubeSat-class hardware. The achieved inference latencies range from 3.20 ms to 30.40 ms. This range demonstrates that all configurations meet or exceed the real-time requirement of 5 FPS for payload control in LEO imaging [27].

B. In-Sensor-based TinyML Results

EuroSAT dataset. The results are shown in Figure 4.

Classification accuracy. MCUNetV1 emerged as the most efficient model, achieving an accuracy of 97.76%. ShuffleNetV2 [28] followed closely with 97.09%. Finally, SqueezeNet scored the lowest overall among the candidates,

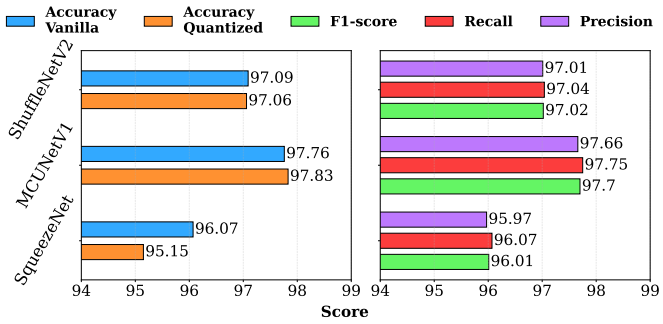

Figure 4: Performance comparison of ShuffleNetV2, MCUNetV1, and SqueezeNet. The left image illustrates the classification accuracy comparisons for both FP32 and INT8 models. The right panel displays the F1-score, Recall, and Precision metrics for the vanilla versions.

although it remained a competitive baseline with an original accuracy of 96.07%.

Reliability metrics. The F1-score, Recall, and Precision achieved balanced performance across all models, indicating no significant class-specific bias.

Impact of quantization. MCUNetV1 and ShuffleNetV2 demonstrated remarkable resilience to bit-width reduction. SqueezeNet proved to be the most susceptible to PTQ, reducing its accuracy to 95.15%.

Hardware-level efficiency. We evaluated the memory footprint in MB and the processing throughput in FPS to quantify real-time performance. Furthermore, to assess energy sustainability, we measured the millijoules consumed per inference and the Giga-Multiply-Accumulate operations per Joule (GMAC/J). On average, our models achieved a processing throughput of 17.40 FPS, maintaining an average latency of 27.43 ms across 100 processed frames. Regarding energy efficiency, the models consumed an average of 14.19 mJ per inference (at an assumed power of 0.247 W). This translates to a computational efficiency of 10.443 GMAC/s and a sustainability metric of 42.26 GMAC/J.

Discussion. By shifting the deep learning workload away from the OBC directly into the image sensor, this study introduces the first optimized in-sensor computing pipeline for EO missions. This paradigm shift effectively resolves two major challenges: it drastically reduces the volume of irrelevant data transmitted over the limited downlink bandwidth, and it eliminates the dynamic power penalty and latency associated with transferring raw Level-0 imagery across the satellite's internal data bus.

IV. Concluding Remarks

The integration of TinyML pipelines can effectively address downlink bandwidth limitations and stringent power and processing constraints while maintaining high energy efficiency for CubeSat missions. Ultimately, shifting critical data processing to the extreme edge of orbit paves the way for sustainable, real-time, autonomous space robotics.

References

[1] O. Kodheli, E. Lagunas, N. Maturo, S. K. Sharma, B. Shankar, J. F. M. Montoya, J. C. M. Duncan, D. Spano, S. Chatzinotas, S. Kisseleff, J. Querol, L. Lei, T. X. Vu, and G. Goussetis, "Satellite Communications in the New Space Era: A Survey and Future Challenges," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 70–109, 2021.

[2] J. D. Liddle, A. P. Holt, S. J. Jason, K. A. O'Donnell, and E. J. Stevens, "Space science with CubeSats and nanosatellites," *Nature Astronomy*, vol. 4, no. 11, pp. 1026–1030, 2020.

[3] N. Crisp, P. Roberts, S. Livadiotti, V. Oiko, S. Edmondson, S. Haigh, C. Huyton, L. Sinpetru, K. Smith, S. Worrall, J. Becedas, R. Domínguez, D. González, V. Hanessian, A. Mølgaard, J. Nielsen, M. Bisgaard, Y.-A. Chan, S. Fasoulas, G. Herdrich, F. Romano, C. Traub, D. García-Almiñana, S. Rodríguez-Donaire, M. Sureda, D. Kataria, R. Outlaw, B. Belkouchi, A. Conte, J. Perez, R. Villain, B. Heißerer, and A. Schwalber, "The benefits of very low earth orbit for earth observation missions," *Progress in Aerospace Sciences*, vol. 117, p. 100619, 2020.

[4] National Aeronautics and Space Administration (NASA), "Earth Observing System Data and Information System (EOSDIS) Handbook," 2018, accessed: 2025-10-25.

[5] A. Babuscia, "Telecommunication Systems for Small Satellites Operating at High Frequencies: A Review," *Information*, vol. 11, no. 5, p. 258, 2020.

[6] Q. Jiang, L. Zheng, Y. Zhou, H. Liu, Q. Kong, Y. Zhang, and B. Chen, "Efficient On-Orbit Remote Sensing Imagery Processing via Satellite Edge Computing Resource Scheduling Optimization," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 63, pp. 1–19, 2025.

[7] A. Furutanpey, Q. Zhang, P. Raith, T. Pfandzelter, S. Wang, and S. Dustdar, "FOOL: Addressing the Downlink Bottleneck in Satellite Computing With Neural Feature Compression," *IEEE Transactions on Mobile Computing*, vol. 24, no. 8, pp. 6747–6764, 2025.

[8] G. Giuffrida, L. Diana, F. de Gioia, G. Benelli, G. Meoni, M. Donati, and L. Fanucci, "CloudScout: A Deep Neural Network for On-Board Cloud Detection on Hyperspectral Images," *Remote Sensing*, vol. 12, no. 14, p. 2205, 2020.

[9] G. Giuffrida, L. Fanucci, G. Meoni, M. Batic, L. Buckley, A. Dunne, C. van Dijk, M. Esposito, J. Hefele, N. Vercruyssen, G. Furano, M. Pastena, and J. Aschbacher, "The Phi-Sat-1 Mission: The First On-Board Deep Neural Network Demonstrator for Satellite Earth Observation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–14, 2022.

[10] National Aeronautics and Space Administration (NASA), "CubeSat Technology Past and Present: Current State-of-the-Art Survey," 2021, accessed: 2025-10-25.

[11] T.-A. Bui, P.-J. Lee, C.-S. Liang, P.-H. Hsu, S.-H. Shiu, and C.-K. Tsai, "Edge-Computing-Enabled Deep Learning Approach for Low-Light Satellite Image Enhancement," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 17, pp. 4071–4083, 2024.

[12] STMicroelectronics, "The STM32N6 Series," https://www.st.com/en/microcontrollers-microprocessors/stm32n6-series.html, accessed: 2025-11-30.

[13] J. Dalbins, K. Allaje, I. Iakubivskyi, J. Kivastik, R. O. Komarovskis, M. Plans, I. Sunter, H. Teras, H. Ehrpais, E. Ilbis, M. Noorma, A. Slavinskis, M. Merisalu, and P. Janhunen, "ESTCube-2: The Experience of Developing a Highly Integrated CubeSat Platform," in *Aerospace Conference (AERO)*, 2022.

[14] Y. M. O. Abbas, E. Edwar, M. A. C. Purio, and A.-H. Jallad, "Onboard Image Classification Unit Implementation for AlAinSat-1 CubeSat," in *IGARSS 2024 - 2024 IEEE International Geoscience and Remote Sensing Symposium*, 2024.

[15] S. Ghatul, B. P. Chandra, S. Jain, B. G. Nair, M. Babu, R. Mohan, M. Safonova, and J. Murthy, "CubeOps: development of an STM32-based on-board computer (OBC) for small satellites and CubeSat missions," in *Space Telescopes and Instrumentation 2024: Ultraviolet to Gamma Ray*, 2024.

[16] M. Eshaq, M. S. Zitouni, S. Atalla, S. Al-Mansoori, and M. Macdonald, "CubeSat Flight Software: Insights and a Case Study," *Journal of Spacecraft and Rockets*, vol. 62, no. 4, pp. 1328–1345, 2025.

[17] STMicroelectronics, "ST Edge AI Developer Cloud," https://stm32ai.st.com/st-edge-ai-developer-cloud/, Accessed: 2025-11-01.

[18] STMicroelectronics, "Neural-ART Processor Unit Tips and Limitations," https://stedgeai-dc.st.com/assets/embedded-docs/stneuralart_operator_support.html, accessed: 2026-01-19.

[19] L. Capogrosso, F. Cunico, D. S. Cheng, F. Fummi, and M. Cristani, "A Machine Learning-Oriented Survey on Tiny Machine Learning," *IEEE Access*, vol. 12, pp. 23 406–23 426, 2024.

[20] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[21] The Linux Foundation, "Open Neural Network Exchange (ONNX)," https://onnx.ai, Accessed: 2025-11-01.

[22] Sony, "IMX500," https://developer.sony.com/imx500, accessed: 2025-11-30.

[23] Sony Semiconductor Solutions, "Model Compression Toolkit (MCT)," https://github.com/SonySemiconductorSolutions/mct-model-optimization, accessed: 2026-03-05.

[24] P. Helber, B. Bischke, A. Dengel, and D. Borth, "EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 7, pp. 2217–2226, 2019.

[25] L. Zhao, P. Tang, and L. Huo, "Feature significance-based multibag-of-visual-words model for remote sensing image scene classification," *Journal of Applied Remote Sensing*, vol. 10, no. 3, p. 035004, 2016.

[26] F. Alam, T. Alam, M. A. Hasan, A. Hasnat, M. Imran, and F. Ofli, "MEDIC: A Multi-Task Learning Dataset for Disaster Image Classification," *Neural Computing and Applications*, vol. 35, no. 3, pp. 2609–2632, 2022.

[27] B. Mellinkoff, M. Spydell, W. Bailey, and J. O. Burns, "Investigation of minimum frame rate for low-latency planetary surface teleoperations," *arXiv preprint arXiv:1706.03752*, 2017.

[28] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.