scientific reports

Check for updates

OPEN

Optimizing deep learning models for on-orbit deployment through neural architecture search

Roberto Del Prete[1,3]✉, Parampuneet Kaur Thind[1,2], Andrea Mazzeo[3], Matthew Whitley[2], Lorenzo Papa[1], Nicolas Longépé[1] & Gabriele Meoni[1]

Advancements in spaceborne edge computing have facilitated the incorporation of Artificial Intelligence (AI)-powered chips into CubeSats, enabling intelligent data handling and enhanced analytical capabilities with greater operational autonomy. This class of satellites faces stringent energy and memory constraints, necessitating lightweight models typically obtained via compression techniques. This paper addresses model compression through Neural Architecture Search (NAS), enabling computational efficiency and balancing accuracy, size, and latency. More specifically, we design an evolutionary-based NAS framework for onboard processing and evaluate it on both burned-area segmentation and classification tasks. The proposed solution jointly optimizes network architecture and deployment for hardware-specific, resource-constrained platforms, with hardware awareness embedded in the optimization loop to tailor network topologies to the target edge computing chip. The resulting models, designed on CubeSat-class hardware-namely the NVIDIA Jetson AGX Orin™ and Intel® Movidius™ Myriad™ X-exhibit a memory footprint below 1 MB, achieving real-time, high-resolution inference in orbit while outperforming handcrafted baselines in terms of latency (3× faster) and maintaining competitive mean Intersection over Union (mIoU). Furthermore, on the classification benchmark conducted on an NVIDIA A100-SXM, our approach attained an atthew Correlation Coefficient (MCC) of 0.974-substantially outperforming the baseline EfficientNet-lite0 (0.902) while achieving a ×47 speedup. These results highlight the framework's scalability, enabling seamless deployment across the spectrum from resource-constrained edge devices to datacenter-grade accelerators, thereby supporting next-generation on-orbit computing architectures.

Earth Observation (EO) satellites are critical for observing Earth's atmosphere, oceans, land, and cryosphere[1], and they underpin research in climate dynamics[2], urbanization[3], and epidemiology[4]. However, conventional ground-based processing pipelines-dependent on post-downlink analysis-are inadequate for real-time applications requiring sub-hour response times. To overcome these limitations, recent efforts have turned to embedding AI and Machine Learning (ML) capabilities directly onboard satellites (Fig. 1). This enables in-situ semantic analysis and intelligent data filtering, reducing transmission of low-value data-such as cloud-covered scenes-and easing downlink constraints[5–7], while satisfying the latency demands of time-critical applications[8]. In parallel, advances in Internet of things (IoT) and edge computing are promoting interdisciplinary convergence and lowering the cost of entry for satellite missions, catalyzing growth in the New Space Economy[9] As AI-driven spaceborne edge computing reshapes the full "acquisition-to-feedback" pipeline[8,10], there is a growing deployment of small satellites equipped with AI-enabled processors[11–15]. Deploying ML models in orbit remains constrained by the harsh spaceborne environment-particularly power limitations, thermal dissipation, and radiation tolerance-which significantly influence the complexity and architecture of models that can be deployed on edge hardware. While future satellite chipsets may boost onboard compute, efficiency remains essential for mission longevity, cost, and reliability[9]. A key metric is the duty cycle-the fraction of time processing is active. Optimizing it requires low-power, time-efficient models suited to intermittent operation constrained by power, thermal, or communication limits.

Model compression and parameter efficiency techniques have thus become indispensable tools for optimizing neural network deployment in resource-constrained environments[16]. These approaches minimize

[1]Φ-lab, European Space Agency (ESA), ESRIN, Via Galileo Galilei, Frascati 00044, Italy. [2]Little Place Labs Ltd, 128 City Road, London EC1V 2NX, UK. [3]Department of Industrial Engineering, University of Naples Federico II, Piazzale Tecchio, 80, Naples 80125, Italy. ✉email: roberto.delprete@esa.int

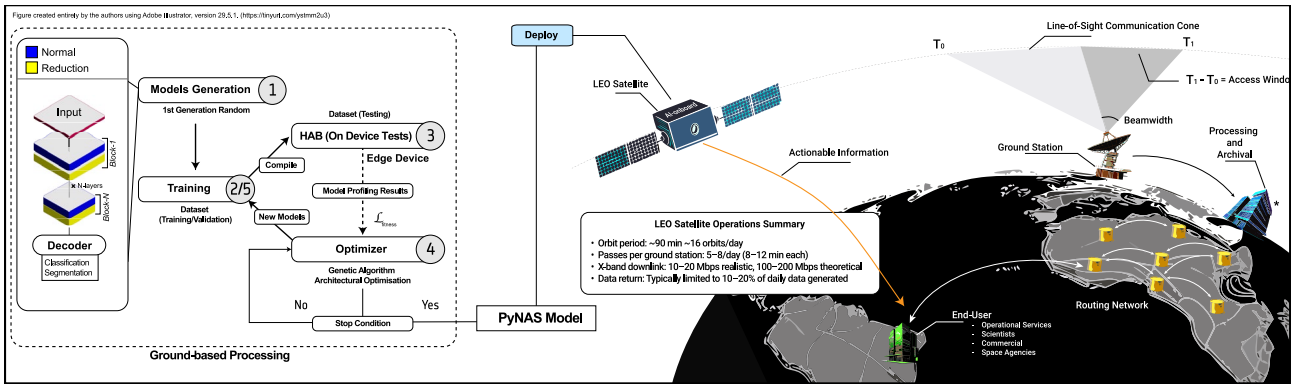


Fig. 1. Block diagram of the proposed hardware-aware framework, which jointly optimizes task-specific performance metrics and device-specific latency. The graphical representation illustrates the AI-enabled Earth Observation (EO) data handling pipeline. In contrast to traditional processing chains-typically constrained by line-of-sight downlink windows of 8–12 min per orbit-the integration of onboard AI significantly reduces data volume by up to 85%, and decreases latency from hours to minutes.

model footprint and computational demand while preserving accuracy, facilitating compliance with real-time processing requirements and stringent hardware limitations. Such efficiencies are crucial given the inherent constraints of spaceborne platforms, where communication bandwidth restrictions impose strict caps on the volume of data and model updates transmissible during brief contact intervals. For example, uplink rates in small satellite missions often limit daily model updates to tens or hundreds of megabytes, a figure further reduced by operational overhead, complicating timely model adaptation to evolving payload conditions or sensor drift[12]. Compounding these challenges is the growing adoption of large foundation models in remote sensing, which, despite their superior performance, present significant computational and memory burdens. Their integration within spaceborne systems demands innovative strategies to balance accuracy, power consumption, and temporal efficiency under strict physical constraints. Ultimately, advancing computational efficiency through model design tailored to optimize duty cycle and resource usage is essential to realize scalable, robust onboard intelligence across diverse satellite architectures. Conventional model compression techniques primarily aim to reduce the size of pre-existing architectures, enhancing efficiency while remaining inherently limited by the original design constraints[17,18]. In contrast, NAS automates the discovery of optimized network topologies and has been successfully applied to both convolutional[19,20] and attention-centric architectures[21,22]. A key advantage of NAS lies in its potential to incorporate hardware-awareness into the optimization loop, explicitly accounting for memory usage, power consumption, quantization-induced performance degradation, and latency constraints. However, while model optimization and NAS techniques have been advanced significantly in the research community, their integration into onboard satellite AI pipelines remains underexplored, particularly under the stringent constraints of spaceborne deployment.

To address this gap, we propose a Genetic Algorithm (GA)-based NAS framework that builds upon the hardware-aware principles introduced in ProxylessNAS[23], extending them to support direct architecture search on the target task and under explicit hardware constraints. By incorporating latency-aware evaluation and resource-bounded architectural generation, our approach is tailored to identify models that not only satisfy predictive performance criteria but also adhere to the inference and energy requirements of edge platforms in orbit-thereby enabling efficient and autonomous onboard decision-making.

Consequently, we make the following contributions: (1) we introduce a hardware-aware NAS framework explicitly designed for edge devices operating onboard satellites; and (2) we conduct a rigorous empirical validation of our methodology on a novel and representative dataset. To the best of the authors' knowledge, conversely to prior works-such as Peng et al.[24], Li et al.[25], and Kadway et al.[26]-our approach is the first to employ a population-based hardware-aware NAS. While these earlier studies explore efficient architecture search strategies or zero-shot scoring methods, none of them jointly optimize for task accuracy and hardware constraints.

The remainder of this paper is organized as follows. The *Background* section reviews prior work on model compression and hardware-aware neural architecture search. The *Methods* section details the proposed NAS framework and its hardware-aware optimization strategy. The *Results* section introduces the datasets developed for benchmarking and presents experimental findings that demonstrate the effectiveness of the approach. Finally, the *Conclusions* section summarizes the contributions and outlines directions for future research.

Background

Onboard satellites-especially CubeSats-computational efficiency is paramount due to strict power, thermal, and memory constraints. Hardware-aware optimization and model compression are key to enabling complex analytics without ground support. The following subsections present two core strategies-model compression and NAS-for efficient AI deployment in spaceborne environments.

Model compression techniques

Model compression transforms large neural network architectures into compact and efficient forms, striving to maintain predictive accuracy while reducing resource requirements. This efficiency is particularly critical when deploying models under stringent memory, power, and bandwidth constraints. Common compression methodologies encompass quantization[27], pruning[28], Knowledge Distillation (KD)[29], and low-rank decomposition[30,31], each offering distinct trade-offs between compression ratio, computational overhead, and predictive fidelity[32–35].

Quantization reduces memory footprint by decreasing numerical precision, commonly converting 32-bit floating-point parameters to formats such as 8-bit integers[27,36]. Approaches include Post-Training Quantization (PTQ), performed post-training, and Quantization-Aware Training (QAT), integrated during training. Despite its storage and computational efficiency, quantization requires careful calibration to balance bitwidth reduction with minimal accuracy degradation due to inherent trade-offs[37,38].

Pruning simplifies network architectures by eliminating redundant or less significant weights and neurons, thereby creating sparser networks. This process directly reduces both memory usage and computational complexity, aiming to preserve original predictive performance[28].

Low-rank decomposition techniques, such as singular value decomposition and advanced tensor factorization methods, approximate weight matrices by decomposing them into products of smaller matrices. These approaches achieve reductions in computational complexity and memory usage, with varied effectiveness depending on the method and application context[30,31,39]. KD leverages knowledge transfer from larger "teacher" networks to smaller "student" architectures. This approach enables compact networks to achieve performance comparable to more complex models across diverse applications. However, KD's efficacy is sensitive to differences in capacity and architecture between teacher and student models and demands meticulous hyperparameter tuning to avoid information loss from softened targets[29,40]. Along the compression techniques, NAS offers more than an alternative, as it can be integrated with quantization-aware training and structured pruning to jointly optimize architecture and weight representations, targeting low-precision accelerators (*e.g.*8-bit integers) for efficient deployment.

Neural architecture search

Rather than compressing an existing model, NAS search for optimal architectures from scratch, providing a proactive approach to balancing model efficiency and predictive capability. NAS formulates the problem as a constrained optimization over the architectural search space $\mathscr{A}$. Specifically, the objective is to identify an architecture $a \in \mathscr{A}$ that minimizes a task-specific loss function $\mathscr{L}(a, \mathscr{D})$, while satisfying resource constraints such as computational cost $\mathscr{C}(a)$. Contemporary NAS approaches employ a variety of empirical strategies to navigate this space efficiently, including learning-based methods and meta-heuristic algorithms. Core strategies can be broadly categorized into GA[41], Reinforcement Learning[42], One-Shot Methods[43], Bayesian Optimization[44,45], and Gradient-Based Approaches[46,47]. The following provides a brief overview of these methods, highlighting their key characteristics. GA evolve a population of candidate architectures through iterative mutation and recombination. Reinforcement Learning-based NAS employs a controller-typically an Recurrent Neural Network (RNN)-to generate architectures, optimizing them via reward signals from performance evaluations. One-Shot methods, such as Differentiable Architecture Search (DARTS)[48] and ProxylessNAS[23], utilize an overparameterized supernet that enables efficient evaluation of sub-architectures without retraining. Bayesian Optimization (BO) leverages probabilistic models to guide exploration toward promising regions of the search space. Gradient-Based methods reformulate architecture search as a continuous optimization problem, enabling the use of gradient descent for efficient exploration. Notably, hybrid strategies often combine One-Shot and gradient-based techniques, as exemplified by DARTS[48] and ProxylessNAS[23], to mitigate the limitations of individual approaches.

It is worth noting that each of these NAS methods offer distinct advantages and challenges. For example, GA and reinforcement learning are well-suited for exploring a wide variety of architectural configurations but are computationally intensive. One-shot methods and gradient-based approaches are efficient due to weight-sharing mechanisms, though this can constrain the performance of individual architectures. BO reduces search time by focusing on promising candidates, albeit at the cost of potentially missing out on high-performing architectures due to biased exploration.

Despite surpassing manual and compressed models in performance and compactness, early NAS methods often decouple accuracy from efficiency, neglecting deployment-critical factors such as memory, latency, power, and radiation resilience-especially vital for spaceborne edge computing. Hardware-aware NAS addresses these limitations by embedding deployment constraints into the search, yielding models suited for real-time, resource-constrained inference. Though gaining traction, its application in Earth Observation is limited, with most efforts focusing on ground-based tasks[24–26,49–51]. Notable onboard attempts include a zero-shot NAS for low-power devices[26] and a reinforcement learning approach for fire detection on nanosatellites[51]. Despite their relevance, these methods face key limitations: restricted search spaces, fixed or heuristic objectives, and limited architectural diversity. Crucially, they overlook accuracy-efficiency trade-offs essential for autonomous, real-time inference in orbit.

Methods

Figure 1 illustrates the developed framework, which can be schematically divided into three key building blocks: Model Generator, Optimizer, and Hardware Awareness Block (HAB).

The Model Generator creates new architectures from the search space, which lists all possible neural network designs to explore during optimization. It provides the basis for generating candidate models. The Optimizer

searches this space to find designs that trade off accuracy and efficiency. It uses algorithms to improve model choices step by step. The HAB adds hardware benchmarking, so chosen designs are tailored to the target device.

Search space

The admissible set of candidate architectures is formally defined as $\mathscr{A}$, from which individual configurations $a \in \mathscr{A}$ are stochastically sampled during the optimization process. Each architecture is instantiated by a *single-path hierarchical generator* that sequentially composes convolutional blocks, as also displayed in Fig. 1. Each block incorporates two modular primitives: a *normal cell*, consisting of standard convolutional operations, and a *reduction cell*, which halves the spatial resolution via max or average pooling. This cell-stacking design paradigm is widely adopted in both handcrafted and NAS-derived architectures. The total number of blocks is uniformly sampled from a discrete interval $[3, M_n]$, where M_n is a tunable upper bound.

Candidate architectures are parameterized by a combination of discrete and continuous variables, including the type and order of layers (e.g., convolutional, pooling, fully connected), as well as hyperparameters such as kernel size, number of filters, and activation function. Each architectural configuration serves as a *genotype*, akin to chromosomes in genetic algorithms, encoding structural elements like skip connections and network depth. This encoding enables expressive and flexible representations within the search space.

Each candidate architecture $a \in \mathscr{A}$ is represented by a structured string called the *architecture code*. For instance, the architecture code `LRr3agn1EPaELco2k3s2p2agn1EPMEHCEE` corresponds to the genotype `["LRr3agn1", "Pa", "Lco2k3s1p1agn1", "PM", "HC"]`, where each segment encodes a functional block. The token `Lco2k3s1p1agn`, for example, specifies a convolutional layer with kernel size 3, stride 1, padding 1, GELU activation, and doubled output channels. This code is parsed into a sequence of components-referred to as the *genotype*-that specify the network's building blocks and their order. This multi-level encoding mechanism-spanning from symbolic strings to interpretable layer definitions-enables expressive model specification and efficient traversal of the architectural search space. Moreover, it provides a robust foundation for mutation and crossover operations in the genetic algorithm, ensuring the generation of structurally valid and hardware-feasible architectures.

To ensure real-time feasibility for embedded platforms, the generator enforces a strict upper bound on model complexity by capping the number of trainable parameters at 10 million. Once instantiated, each candidate architecture $a \in \mathscr{A}$ is trained via supervised learning and evaluated using a hardware-aware fitness function.

Optimization strategy

As one of the first contributions tailored to on-orbit satellite deployment, the architecture search strategy is formulated using a GA. This methodological choice is motivated by two key considerations: (1) the widespread success of evolutionary approaches in NAS, and (2) their inherent ability to optimize non-differentiable, discrete objective functions without relying on continuous relaxations. The optimization process operates over a fixed number of generations G_n, with each generation maintaining a population of P_s candidate architectures. The evolutionary cycle iteratively refines the population through a balance of exploration and exploitation.

Two primary genetic operators drive population evolution. First, *mutation*, applied at a probability $M_f(\%)$, introduces stochastic variability by randomly replacing one layer in an architecture with a structurally distinct alternative from the predefined search space. This mechanism ensures continual architectural innovation and prevents premature convergence. Second, *single-point crossover* is applied to the top-performing $M_p(\%)$ of the population (the mating pool), ranked by fitness. For each selected parent pair, a random crossover point is sampled, and the architectural components before and after this point are exchanged, producing two offspring. This recombination process promotes the inheritance of beneficial architectural traits, accelerating convergence toward high-performing solutions.

To promote convergence toward high-quality solutions while preserving exploratory capability, the evolutionary process integrates three key mechanisms. First, elitism is applied: the top K_{best} architectures, ranked by fitness, are retained across generations, ensuring monotonic performance improvement.

Second, diversity injection mitigates premature convergence. At each generation, n_{random} novel architectures are uniformly sampled from the search space, enabling exploration of underrepresented regions that may elude incremental refinement.

Third, fitness-based selection guides genetic operations. Architectures are chosen with probabilities proportional to their multi-objective fitness scores, favoring those that best trade off accuracy and efficiency. To calculate the fitness, we adopt a weighted sum with exponential penalty, formulated as follows:

$$\mathscr{L}_{fitness} = \alpha \cdot \widehat{\text{fps}} + \beta \cdot \text{Metric} \cdot e^{\gamma \cdot \text{Metric}} \tag{1}$$

where α, β, and γ are scalar weights that balance the contributions of speed and accuracy, enabling task-specific trade-offs during optimization. In this experiment we set $\alpha, \beta, \gamma = 1$ to balance the influence of speed and accuracy, noting that the exponential term increases the weight of higher accuracy values.. The inference speed is normalized against a target value of 120 Frames Per Second (FPS) ($\text{fps}_{\text{target}} = 120$), yielding $\widehat{\text{fps}} = \frac{\text{fps}}{120}$, a dimensionless metric adjustable via a hyperparameter to reflect deployment needs. Finally, the model accuracy is captured via a task-specific Metric, with exponential weighting $e^{\gamma \cdot \text{Metric}}$ to emphasize high-performing architectures and penalize low-performance cases. In our case we use mIoU as our accuracy metric.

Hardware-awareness

After each training cycle, the training server interfaces with a designated device to collect performance feedback. Specifically, the NAS process performs inference on the target deployment environment to evaluate system-level

metrics that inform the fitness function. If required by the device, model quantization and hardware-specific optimizations are integrated into the NAS workflow during the compilation phase, ensuring that resulting architectures are not only efficient but also compatible with deployment constraints. This feedback loop enables the search process to jointly optimize for both predictive accuracy and hardware-aware metrics-such as inference latency and memory access cost-ensuring that the selected architectures meet the stringent requirements of real-time, resource-constrained onboard applications.

Results

Data

We evaluated the proposed NAS approach on two separate tasks, i.e., thermal hotspots classification and burnt area segmentation.

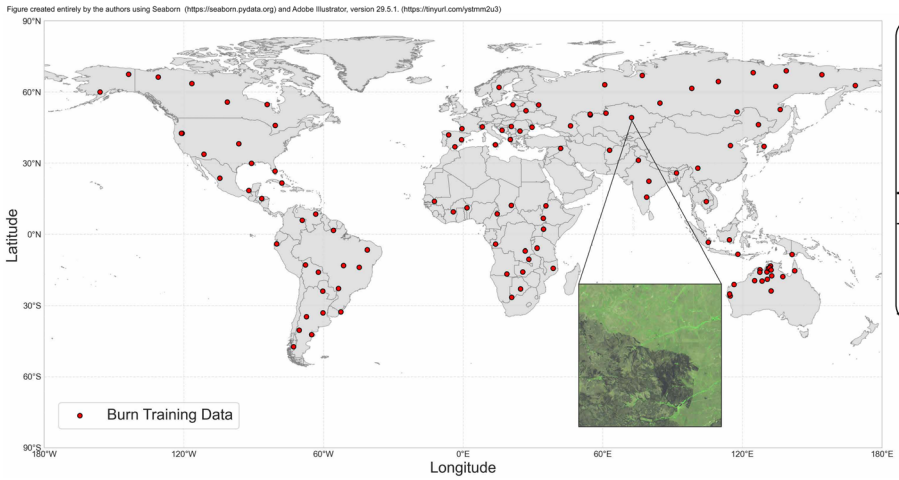
Burnt area segmentation

We developed a novel dataset with near-global coverage giving particular attention to class balance across training, validation, and test splits to ensure robust learning and evaluation. The legend in Figure 2 shows how class distributions are carefully structured to avoid bias, promoting generalization across diverse geospatial and ecological conditions.

The burned area dataset consists of high-resolution, multispectral Sentinel-2 imagery curated for post-fire analysis, supporting accurate delineation of burned regions across varied ecological and climatic settings. While Sentinel-2 offers 10 m spatial resolution for most bands, we simulate the higher-resolution and sensor-specific characteristics of Φ-sat-2 to support realistic onboard deployment scenarios. This simulation bridges the gap between freely available satellite data and the constraints and opportunities of in-orbit sensing systems like Φ-sat-2. The dataset comprises 115 fire events geographically distributed across North America (17), South America (17), Africa (17), Europe (18), Asia (29), and Australia (17), thereby ensuring broad geographic and climatic coverage. By encompassing fire regimes from boreal, temperate, and tropical ecosystems, the dataset supports comprehensive analyses of fire severity, post-fire vegetation dynamics, and ecosystem resilience. The inclusion of spatially and temporally diverse fire events enhances the generalization capability of learned models, facilitating robust performance across a wide range of environmental conditions.

To approximate Φ-sat-2 onboard observations, we applied a dedicated simulation pipeline to the Sentinel-2 L1C inputs. This included selection of key bands (B02, B03, B04, B08, B05, B06, B07) and derivation of cloud, cloud shadow, and cirrus masks from the Scene Classification Layer (SCL). Solar geometry and irradiance metadata were used to compute radiances, from which a synthetic pan-chromatic band was derived. Sentinel-2 imagery was then resampled to 4.75 m resolution, aligning with Φ-sat-2's imager. To reflect in-orbit sensor characteristics, the imagery was also degraded with simulated band misalignment, signal-dependent noise profiles (SNR), and modulation transfer function (MTF) blur. In some cases, simulated L1C reflectances were recalculated from radiance. Finally, image chips of size 256×256 were tiled and saved with corresponding masks, producing an AI-ready dataset closely emulating Φ-sat-2's onboard acquisition conditions.

To improve classification reliability and address spectral ambiguities, annotations are structured into four classes: *Background*, *Burned Areas*, *Clouds*, and *Waterbodies*-with the latter two explicitly included to mitigate occlusion effects and reduce misclassification. The explicit labeling of *Clouds* and *Waterbodies* is particularly critical, as these features frequently obscure terrestrial surfaces in optical satellite imagery. Cloud cover, characterized by high spatial and temporal variability, introduces discontinuities that hinder burned area detection, while waterbodies exhibit spectral signatures that may be erroneously classified as burn scars if not properly distinguished. By incorporating these additional classes, the dataset substantially improves segmentation robustness, reducing errors associated with atmospheric interference and surface-level ambiguities.



Class (# of images)	Test (2000)	Val. (2335)	Train (5449)
Background	20.0	20.4	59.7
Burned Area	19.8	20.1	60.1
Cloud	19.4	20.1	60.5
Water Body	20.8	19.4	59.8

Class	Pixel Proportion	Time Coverage
Background	25.9%	(2015–2023)
Burned Area	24.7%	(2015–2021)
Cloud	23.2%	(2018–2023)
Water Body	26.3%	(2018–2023)

Fig. 2. Overview of the dataset employed in this study showing the spatial distribution of sampled data points with an example tile. The legend displays the class-wise percentage composition.

Thermal hotspots classification

To showcase the generality of our approach on a different task, we employ the "end2end" dataset from our previous work[6], specifically designed to enable the development of models for onboard, real-time thermal anomaly detection. Unlike conventional imagery that rely on atmospheric correction or fine geometric registration, the dataset preserves the characteristics of raw Sentinel-2 granules, using only pre-computed coarse shifts to align bands B11 and B12 with B8A. This design choice reflects the constraints of in-orbit processing, where minimal pre-processing is essential to meet latency and computational requirements. Each granule, approximately 1152×1296 pixels, was subdivided into 256×256 patches, and patches containing at least nine hotspot pixels within a THRawS bounding box were labeled as "events," with all others labeled "non-events." Labels were further refined through visual inspection to ensure reliability. The resulting collection comprises 5033 patches, with a strong imbalance between event (394, 7.8%) and non-event (4636, 92.2%) classes, thereby reflecting the rarity of thermal anomalies in real-world monitoring scenarios. To approximate operational deployment, the dataset was split into a geographically stratified manner, ensuring that training and testing samples originate from distinct regions while maintaining class proportions.

Evaluation metrics

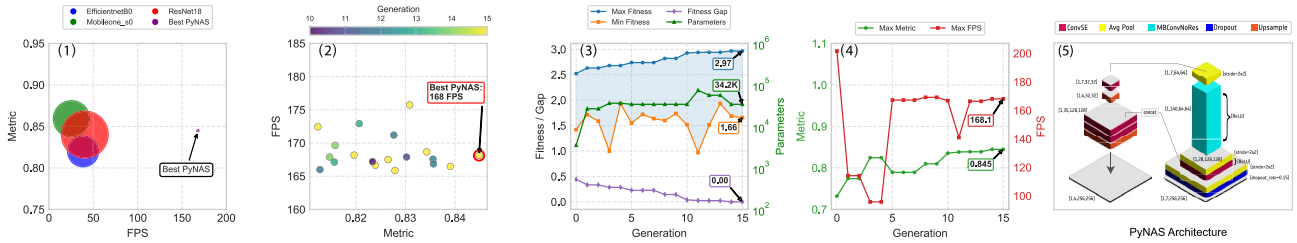
The metric chosen for evaluating model performance is the mIoU, a standard metric for semantic segmentation. It is defined as the average IoU computed across all semantic classes: $\mathrm{IoU}_c = \frac{|S_p^c \cap S_g^c|}{|S_p^c \cup S_g^c|}, \quad \mathrm{mIoU} = \frac{1}{C}\sum_{c=1}^{C} \mathrm{IoU}_c$, where S_p^c and S_g^c denote the predicted and ground truth masks for class c, and C is the number of classes. The mIoU quantifies spatial overlap and is particularly effective for evaluating segmentation performance across heterogeneous class distributions. For classification, the MCC is selected as metric providing a balanced evaluation even under class imbalance problems. The MCC is defined as $\mathrm{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$, where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively.

Experiments

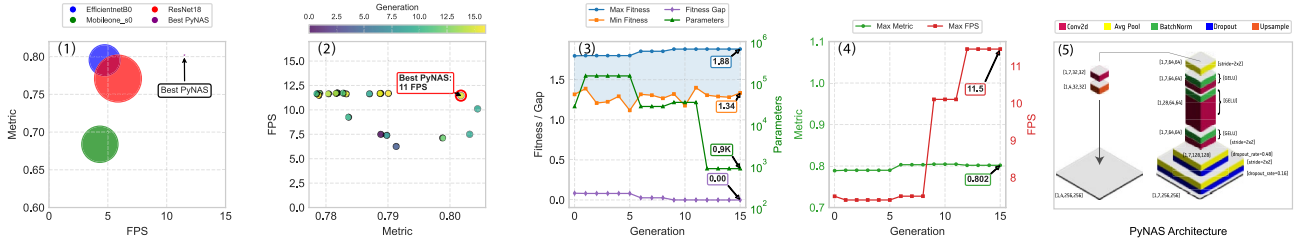
All models were trained on an Amazon EC2 instance of type `g4dn.12xlarge`, equipped with four NVIDIA T4 Graphics Processing Units (GPUs), 48 vCPUs, and 192 GB of RAM, running Ubuntu 24.04 LTS (Noble Numbat). The software stack included CUDA 12.4 and PyTorch 2.6.0. To ensure a fair and reproducible comparison across candidate architectures, no hyperparameter tuning was performed. Focal Loss[52] was used during training to address class imbalance in segmentation tasks. This improves sensitivity to hard-to-segment regions and sharpens boundary delineation. To assess deployment feasibility under spaceborne constraints, we evaluate our NAS framework on two complementary edge platforms: the NVIDIA Jetson AGX Orin™[53] and the Intel® Movidius™ Myriad™ X[54]. These devices exemplify distinct trade-offs in computational capability, energy efficiency, and integration complexity, providing a comprehensive perspective on onboard AI processing. The NVIDIA Jetson AGX Orin™ delivers up to 275 TOPS within a 15–60 W envelope, supporting 32-bit and mixed-precision inference via CUDA and TensorRT. A variant, the Orin NX, has been selected for space deployment aboard SpaceX's Transporter-11, equipped with radiation shielding[55]. The Myriad™ X offers over 1 TOPS at 2 W, with 16-bit fixed-point support and a dedicated Neural Compute Engine. It has demonstrated in-orbit capability on D-Orbit's Wild Ride ION mission for onboard Earth observation[56]. In addition, we evaluated our framework on the NVIDIA A100-SXM (300 W), a datacenter-grade accelerator delivering up to 312 TFLOPS of mixed-precision performance and optimized for large-scale AI workloads. The contrast between 32-bit and 16-bit inference regimes underscores the breadth of our evaluation, showcasing the NAS framework's capacity to adapt across heterogeneous precision constraints and deployment scenarios. We note that Jetson AGX Orin experiments used 32-bit precision despite native 16- and 8-bit support, as our goal was to assess NAS robustness across hardware rather than optimize a single device. On both devices, each model required about 2 minutes per generation, with the full NAS procedure lasting roughly 48 hours for the segmentation task and about 4 hours for the classification task, subject to dataset size and parameter choices.

Segmentation task

Figure 3a displays results for the NVIDIA Jetson AGX Orin™. Particularly, the Fig. 3a(1) shows that the model discovered by our approach achieves a competitive mIoU while delivering an exceptional inference throughput-approximately $3\times$ faster than baseline architectures, *i.e.*as MobileOne-S0[57], EfficientNet-B0[19], and ResNet-18[58]. Indeed, the model, highlighted by an arrow, occupies the right segment of the metric–FPS space, decisively dominating the performance–efficiency trade-off. ResNet-18, with 18 layers and approximately 11.7M parameters, achieves a mIoU of 0.840 at 40.4 FPS (fitness = 2.28). Despite its accuracy, it exhibits substantially higher latency and complexity compared to our model, which matches its performance while using only 34.2 K parameters. EfficientNet-B0 achieves a lower mIoU of 0.820 and an inference speed of 38.5 FPS (fitness = 2.18), whereas MobileOne-S0 yields a higher accuracy of 0.859 but at a lower throughput of 25.4 FPS (fitness = 2.24). Despite these trade-offs, none of the baselines jointly optimize for both accuracy and latency as effectively as the proposed architecture. The Pareto front, shown in Fig. 3a(2), comprises the top 20 architectures output of the NAS optimization. The graph outlines how the final optimized model reaches the mIoU of 0.845 while delivering an exceptional inference throughput of 168.1 FPS (fitness = 2.97). Notably, the evolutionary algorithm exhibits consistent convergence across successive generations, attaining a peak fitness value of 2.97 by generation 15 (see Fig. 3a(3))—higher than any of the baseline models. This convergence is further substantiated by a pronounced reduction in the fitness gap, indicative of diminished variance and the emergence of dominant, high-performing architectural configurations. Finally, the Fig. 3a(4) further illustrates the joint progression of maximum mIoU and FPS across generations.



(a) NVIDIA Jetson AGX Orin™



(b) Intel® Movidius™ Myriad™ X

Fig. 3. Summary of the NAS process on the segmentation task. Subfigures illustrate: (1) performance–latency–complexity trade-offs of discovered models compared to state-of-the-art baselines (ResNet-18[58], Mobileone-S0[57], EfficientNet-B0[19]; bubble size denotes parameter count); (2) Pareto front (top 20 models) showing the trade-off between segmentation accuracy (mIoU) and inference speed (FPS)–the deployment-selected architecture is indicated by arrow; (3) Fitness progression over generations, including maximum, minimum, and gap values, along with parameter count evolution of the top-performing model; (4) trajectory of maximum mIoU and FPS across generations, demonstrating joint optimization; (5) architecture discovered by our framework. NAS was performed over 15 generations with a population size of 50, training each model for 10 epochs. In each generation, the top 50% of models composed the mating pool (mutation rate: 0.2), with 10 randomly initialized architectures injected per generation. The highest-performing model was preserved to ensure elitism.

Similar considerations could be applied for the NAS optimization on the Movidius™ Myriad™ X, reported in Fig. 3b. A critical distinction in this context is that all models undergo 16-bit quantization and hardware-specific compilation prior to deployment. The performance variability observed across architectures highlights the absence of hardware-awareness in baseline models, which are not tailored to the operational constraints of the Myriad™ X accelerator. The bubble graph in Fig. 3b(1) shows how our discovered architecture maintains its trade-off advantage over the baseline models. EfficientNet-B0[19] achieves a mean IoU of 0.795 at 4.73 FPS (fitness = 1.80). ResNet-18[58] follows with an accuracy of 0.770 and 5.86 FPS (fitness = 1.72), while MobileOne-S0[57] exhibits the poorest performance on this platform, attaining only 0.683 accuracy and 4.32 FPS (fitness = 1.39).

These results highlight the sensitivity of baseline architectures to hardware constraints and PTQ. Specifically, this pronounced drop in segmentation accuracy for baseline models on the Myriad™ X can be attributed to several architectural mismatches between these models and the hardware's constraints. In particular, ResNet-18 and MobileOne-S0 include layers and operations that are either unsupported or sub-optimally executed on the Myriad™ X. These include large-kernel convolutions (e.g., 7×7 in ResNet-18), unoptimized skip connections, and dynamic activation patterns such as Squeeze-and-Excitation (SE) blocks in MobileOne. Furthermore, the Myriad™ X relies on floating point 16 precision and static memory allocation, making it particularly sensitive to models trained under floating point 32 assumptions or those involving extensive branching and memory reuse. Unlike our proposed architectures-which co-evolved with deployment constraints in mind-baseline models were not designed with the Myriad™ X's compilation and runtime pipeline in consideration. This leads to reduced throughput and degraded inference fidelity after quantization. These findings reinforce the critical importance of hardware-aware design in edge AI pipelines. Even on this constrained platform, our model maintains nearly a 3× speedup over all tested alternatives, confirming its superior hardware generalizability and robustness. This behavior is further highlighted in Fig. 3b(2), where our proposed solution sustains its Pareto dominance even with drastically fewer parameters-less than 1 K-compared to multi-million parameter baselines. In contrast to the Jetson AGX Orin™, the Myriad™ X's performance saturates quickly: further shrinking of model size along the Pareto front offers diminishing returns. This is evident in the altered shape of the fitness progression curve in Fig. 3b(3), which flattens earlier due to hardware-imposed throughput limitations. Fig. 3b(4) illustrates the joint evolution of accuracy and speed. This reflects the effective adaptation of the algorithm to hardware-specific restrictions, particularly in this regime where the latency target of 120 FPS is not attainable. In this scenario, the optimization process places increased emphasis on minimizing inference time, as evidenced by the steep slope of the FPS progression curve. Since further gains in mIoU are limited and latency improvements occur within a relatively narrow performance envelope, the resulting evolution of the

fitness function exhibits a flatter trajectory compared to that of the Jetson AGX Orin™. This behavior confirms the algorithm's sensitivity to hardware-imposed ceilings and its capacity to prioritize latency-aware optimization in low-throughput, resource-constrained environments.

From a qualitative perspective, the Fig. 4 presents a comparison of segmentation masks generated by baseline models (EfficientNet-B0, MobileOne-S0, ResNet-18) and those discovered through our NAS framework (The reader is referred to Appendix B for a comprehensive account of additional qualitative and quantitative comparisons). The results are evaluated against high-resolution ground truth annotations across four representative scenes. The optimized architectures (PyNAS for Myriad™ X and for Jetson AGX Orin™) consistently produce segmentation maps with high spatial fidelity, matching the same, if not better, level of accuracy of the baseline models. Indeed, both the Jetson AGX Orin™- and Myriad™ X-optimized variants yield clean predictions with sharp object contours. Notably, the PyNAS (Jetson AGX Orin™) model achieves superior structural agreement with the ground truth across all samples, particularly in complex scenes of the second and fourth row, involving small, fragmented regions in the top area of the sample. These observations again validate the effectiveness of our approach in discovering architectures that match, if not surpass, manual baselines but with a significant higher computational efficiency. The consistency across hardware targets underscores the robustness and adaptability of the proposed methodology for real-world remote sensing applications.

We finally note that on both devices, each model generation required approximately 2 min per generation, and the complete NAS procedure took about 48 h. These timings may vary depending on the dataset size and the chosen NAS parameters.

Classification task

To showcase the potential of future on-orbit datacenters, as envisioned by European Space Agency (ESA) to enable AI-powered space missions and agile in-orbit applications[59], we also evaluate our approach on a high-performance device, the NVIDIA A100-SXM (300 W). While not currently certified for space deployment due to power and radiation constraints, this experiment illustrates the performance envelope achievable under unconstrained computational conditions, offering insights into the capabilities that next-generation cognitive cloud infrastructures in space could unlock. The results for the classification task are reported in Fig. 5, where our model achieved a MCC of 0.974 and an inference throughput of 8555 FPS. For comparison, the best model in Meoni et al.[6] (EfficientNet-lite0) reached a maximum MCC of 0.902 with 187 FPS, despite extensive experimentation with different learning policies and hyperparameter tuning. This represents an absolute accuracy gain of ≈+8% in MCC, while delivering more than a 47-fold increase in inference speed. It is worth emphasizing that the model of Meoni et al.[60] is already a well-engineered architecture tailored to operate on edge devices, making it a particularly strong baseline. The fact that our solution not only surpasses it in accuracy but also dramatically reduces latency suggests that the proposed hardware-aware search framework does not merely compress existing designs but is capable of discovering fundamentally more efficient architectures. This points to a broader implication: hardware-aware NAS has the potential to bridge the longstanding gap between model compactness and predictive fidelity, a trade-off that has traditionally constrained spaceborne AI applications. The Pareto front output of the evolutionary search is displayed in Fig. 5a while Fig. 5b highlights improvements in MCC and inference speed. Beyond quantitative metrics, the Fig. 5c presents representative inference examples confirming the reliability of the outputs. Concluding the panel, Fig. 5d depicts the PyNAS architecture, whose depth of five was selected by the more powerful available computational resources. Beyond the raw performance figures, these results raise two important reflections. First, they demonstrate the adaptability of our approach across heterogeneous hardware platforms, from highly constrained edge devices to powerful datacenter-grade accelerators. This adaptability suggests that the same methodology could be flexibly repurposed for hybrid Earth–space architectures, where initial training or adaptation occurs on the ground with high-performance GPUs, followed by deployment of compressed and specialized models on orbit. Second, they highlight that

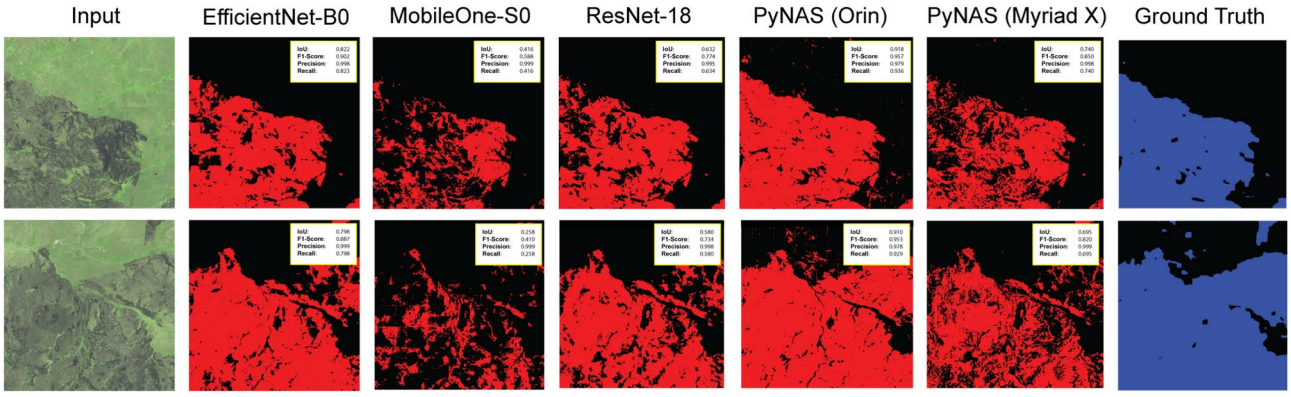


Fig. 4. Qualitative segmentation comparison of segmentation masks produced by various models-EfficientNet-B0, MobileOne-S0, ResNet-18, PyNAS (NVIDIA Jetson AGX Orin™), and PyNAS (Intel® Movidius™ Myriad™ X)-alongside the input image and ground truth. The tiles used have been gathered on 8 August 2019 in northeastern Kazakhstan (48.83°–49.53° N, 71.96°–72.25° E). Each tile spans 1.8 × 1.8 km.

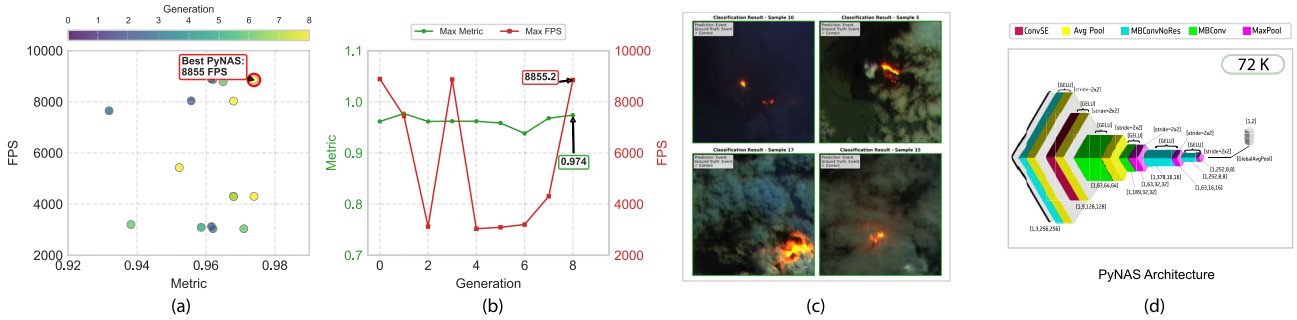


Fig. 5. Summary of the NAS process on the classification task. Subfigures illustrate: (**a**) Pareto front (top 20 models) showing the trade-off between classification accuracy (MCC) and inference speed (FPS)—the deployment-selected architecture is indicated by arrow; (**b**) trajectory of maximum MCC and FPS across generations, demonstrating joint optimization; (**c**) representative inference results on the test set; (**d**) architecture discovered by our framework. NAS was performed over 8 generations with a population size of 50, training each model for 15 epochs. In each generation, the top 50% of models composed the mating pool (mutation rate: 0.2), with 10 randomly initialized architectures injected per generation. The highest-performing model was preserved to ensure elitism.

future advancements in space-qualified hardware may unlock even greater potential for onboard autonomy: as radiation-hardened high-performance GPUs become available, the efficiency gains we observe on high performance chips could be directly translated into operational benefits, such as lower duty cycles, reduced downlink requirements, and enhanced mission resilience.

Conclusion

This study introduces a hardware-aware NAS framework tailored specifically for satellite-based edge processing constraints. By integrating evolutionary optimization with real-time latency profiling on heterogeneous hardware platforms-NVIDIA Jetson AGX Orin™ and Intel® Movidius™ Myriad™ X-we have shown that the proposed approach significantly surpasses state-of-the-art handcrafted architectures in terms of the performance–efficiency trade-off. Additionally, we showcase its potential for future cognitive cloud computing infrastructures in space, aligning with emerging initiatives on on-orbit datacenters and AI-enabled space missions.

Specifically, on the NVIDIA Jetson AGX Orin™, our optimal architecture achieved a remarkable mIoU of 0.845 at 168.1 FPS with merely 34.2K parameters, substantially exceeding the efficiency of MobileOne-S0 (0.859 mIoU at 25.4 FPS) and ResNet-18 (0.840 mIoU at 40.4 FPS), both of which surpass millions of parameters. Similarly impressive results were observed on the Intel® Movidius™ Myriad™ X, where NAS-derived architectures achieved 0.802 mIoU at 11.5 FPS with fewer than 1K parameters, demonstrating extraordinary compactness and robust quantization resilience. Beyond segmentation, the classification task further highlighted the framework's efficiency: on the NVIDIA A100-SXM, our model achieved a MCC of 0.974 at 8555 FPS, compared to the baseline model (EfficientNet-lite0) 0.902 at 187 FPS, representing an ≈+8% accuracy gain and a 47-fold increase in inference throughput. These results, obtained across edge and datacenter-grade platforms, emphasize the scalability and versatility of the proposed methodology.

The fitness convergence analysis further confirmed stable evolutionary optimization, clearly evidencing simultaneous improvements in accuracy and latency. Additionally, qualitative assessments validated the structural fidelity and generalization capability of the segmentation models. The consistency of these results across multiple hardware platforms underscores the effectiveness and reliability of our hardware-in-the-loop methodology.

Building on these findings, while our current results are simulation-based, future work aims to explore real on-orbit deployment. This includes evaluating the approach on suitable hardware platforms to assess its robustness and feasibility under operational satellite conditions. Future directions will also explore multi-objective optimization strategies to include energy efficiency and radiation resilience considerations, and broaden framework applicability to complementary tasks such as anomaly detection or onboard compression. Detailed speed–accuracy trade-offs under reduced-precision quantization are also necessary studies. A systematic comparison of NAS approaches constitutes another key research avenue. Such comparison, while currently constrained by computational resources, will be prioritized as part of our ongoing and future research efforts to ensure a fair and comprehensive assessment of NAS methods for satellite edge computing scenarios. Furthermore, distillation of geospatial foundation models, while not empirically substantiated in the current study, may offer promising opportunities; its potential utility and alignment with satellite-edge processing constraints warrant careful future investigation.

Data availability

All relevant data supporting the findings of this study are either included in the manuscript and supplementary materials or are publicly available in the following repository: https://huggingface.co/datasets/ESA-PhiLab-Edge/LPL-Burned-Area-Seg. Our implementation is available at :https://github.com/ESA-PhiLab/pynas

Appendix A—GA-based NAS Loop

The following Algorithm 1 describes a hardware-aware NAS procedure designed for deployment on edge devices. The search process begins with the random generation of candidate models and progressively shifts towards the optimization of architectures informed by device-specific profiling results. Each model is trained and subsequently profiled on the target hardware to capture critical performance indicators, including latency, accuracy, memory usage, and energy consumption. These results update the optimizer, guiding the generation of architectures towards solutions that achieve the desired trade-off among performance metrics. In this work, the iterative loop terminates once a *generation-based stop condition* is satisfied. Specifically, the procedure is bounded by a maximum number of generations $N_{\max}$. Denoting by $t \in \mathbb{N}$ the current generation index, the adopted stop condition is formally expressed as

$$\mathrm{StopCondition}(t) \;=\; \big(t \geq N_{\max}\big).$$

This choice ensures that the search complexity remains explicitly bounded, while still allowing the optimizer to progressively refine candidate architectures over successive generations.

Although the generation-based rule is employed in this paper, an alternative criterion can be considered in scenarios where resource efficiency and accuracy are paramount. Let M denote a candidate model and let $J(M)$ be a scalarised objective function that aggregates accuracy and resource-related costs. A common scalarisation is

$$J(M) \;=\; \alpha \cdot \mathrm{Acc}(M) - \beta \cdot \mathrm{Cost}(M),$$

where $\mathrm{Acc}(M)$ denotes an accuracy measure (e.g., mean IoU for segmentation or top-1 accuracy for classification), $\mathrm{Cost}(M)$ encodes resource usage (latency, memory, or energy), and $\alpha, \beta > 0$ are weighting factors that define the trade-off.

Given a predefined threshold $\tau \in \mathbb{R}$, a performance-based stop condition can then be expressed as

$$\mathrm{StopCondition}(M) \;=\; \big(J(M) \geq \tau\big).$$

The generation-based formulation guarantees tractable search and was therefore chosen in this study, while the performance-based formulation remains a flexible alternative for contexts in which satisfying strict deployment requirements is prioritised.

```
procedure NeuralArchitectureSearch
    iteration ← 0
    while not StopCondition(iteration) do
        if iteration = 0 then
            models ← GenerateRandomModels()
        else
            models ← GenerateOptimizedModels(profile_results)
        end if
        for all model ∈ models do
            Train(model)
            profile_results ← Profile(model, edge_device)
            UpdateOptimizer(profile_results)
        end for
        iteration ← iteration + 1
    end while
    return BestModelFound()
end procedure
```

Algorithm 1. Hardware-aware neural architecture search loop.

```
function StopCondition(iteration)
    if iteration ≥ N_max then
        return true                ▷ Reached maximum number of generations
    else
        return false
    end if
end function
```

Algorithm 2. Stop condition.

Appendix B—Detailed model comparison

The Table 1 reports the principal computational and storage metrics: number of trainable parameters, FLOPs, MACs, estimated memory usage, and model size under FP32 representation. The NAS-derived networks Myriad and Orin achieve negligible parameter count (less than 0.05 million) and extremely low FLOPs (below 0.5 billion), which translates into a storage footprint close to zero and very fast inference. In contrast, the ResNet-18 U-Net is the heaviest model in this comparison, with approximately 14.3 million parameters and 11.3 billion FLOPs, yielding the largest memory footprint of about 178 MB. It provides strong feature extraction but at a substantial computational cost. The EfficientNet-UNet lies in an intermediate regime, with 6.25 million parameters and 5.1 billion FLOPs, balancing compactness and accuracy by means of MBConv and squeezeâ€“excitation blocks. MobileOne-UNet, despite having fewer parameters than ResNet-18, shows the highest activation memory usage (about 455 MB). This phenomenon occurs because of deeper convolutional stacks and wide intermediate feature maps, which increase runtime memory consumption. We note that this trade-off between parameter count and activation memory is not trivial: MobileOne demonstrates that even with a moderate number of parameters, runtime memory can remain high due to activation-heavy design choices.

Concerning repeceptive fields, NAS-derived networks (Myriad, Orin) exhibit shallow and compact receptive fields consistent with their extremely low parameter count and FLOPs, reflecting limited representational capacity (see Fig. 6). In contrast, ResNet-18 U-Net achieves the broadest receptive field due to its deeper residual hierarchy, albeit at high computational and memory cost. EfficientNet-B0 provides a more balanced receptive field expansion through MBConv and squeezeâ€“excitation modules, while MobileOne-S0, despite its moderate parameter count, attains wide receptive fields via deeper convolutional stacks, which drive its unusually large activation memory usage.

Model	Params (M)	Depth	Layers*	FLOPs (G)	MACs (G)	Memory (MB)	Model size (MB)
PyNAS (Myriad)	0.0009	3	10	0.0061	0.0030	6.10	0.003
PyNAS (Orin)	0.034	4	14	0.862	0.431	$\sim$ 7.00	0.13
ResNet-18	14.34	5	32	11.322	5.661	178.21	57.36
EfficientNet-B0	6.25	5	93	5.123	2.561	147.83	23.9
MobileOne-S0	8.59	5	202	9.524	4.762	455.62	34.4

Table 1. Comparison of U -Net based segmentation models with different encoder backbones. Columns report parameter count (in millions), encoder depth (number of stages), number of RF-affecting layers, FLOPs, MACs, estimated memory footprint, and model size on disk assuming FP32 precision. *Layers counted include `nn.Conv2d`, `nn.MaxPool2d`, `nn.AvgPool2d`, `nn.ConvTranspose2d`, `nn.Upsample`, and `nn.Dropout`.

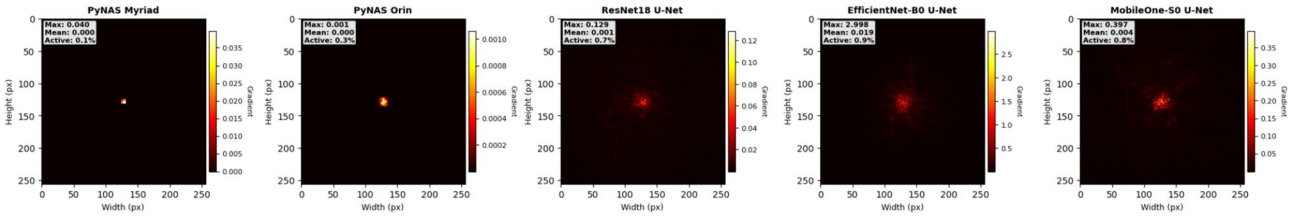


Fig. 6. Comparison of the receptive fields across U-Net based models with different encoder backbones.

Received: 26 May 2025; Accepted: 22 September 2025
Published online: 29 October 2025

References

1. Emery, W. & Camps, A. *Introduction to Satellite Remote Sensing: Atmosphere, Ocean, Land and Cryosphere Applications* (Elsevier, 2017).
2. Yang, J. et al. The role of satellite remote sensing in climate change studies. *Nat. Clim. Change* **3**, 875–883 (2013).
3. Patino, J. E. & Duque, J. C. A review of regional science applications of satellite remote sensing in urban settings. *Comput. Environ. Urban Syst.* **37**, 1–17 (2013).
4. Goetz, S., Prince, S. & Small, J. Advances in satellite remote sensing of environmental variables for epidemiological applications. *Adv. Parasitol.* **47**, 289–307 (2000).
5. Tuia, D. et al. Artificial intelligence to advance earth observation: A review of models, recent trends, and pathways forward. *IEEE Geosci. Remote Sensing Magazine*. https://doi.org/10.1109/MGRS.2024.3425961 (2024).
6. Meoni, G. *et al.* Unlocking the use of raw multispectral earth observation imagery for onboard artificial intelligence. *IEEE J. Selected Topics Appl. Earth Observations Remote Sensing*. (2024).
7. Rijlaarsdam, D. et al. The next era for earth observation spacecraft: An overview of cognisat-6. *IEEE J. Selected Topics Appl. Earth Observations Remote Sensing* **18**, 2450–2463. https://doi.org/10.1109/JSTARS.2024.3509734 (2025).
8. Růžička, V. et al. Ravæn: Unsupervised change detection of extreme events using ml on-board satellites. *Sci. Rep.* **12**, 16939 (2022).

9. Paravano, A., Locatelli, G. & Trucco, P. What is value in the new space economy? The end-users' perspective on satellite data and solutions. *Acta Astronautica* **210**, 554–563 (2023).
10. Del Prete, R. *et al.* Enhanced maritime monitoring via onboard processing of raw multi-spectral imagery by deep learning. In *IGARSS 2024-2024 IEEE International Geoscience and Remote Sensing Symposium*, 1713–1717 (IEEE, 2024).
11. Schöttl, F., Spichtinger, A., Franquinet, J. & Langer, M. Real-time on-orbit fire detection on forest-2. In *IGARSS 2024-2024 IEEE International Geoscience and Remote Sensing Symposium*, 2360–2364 (IEEE, 2024).
12. Longépé, N. et al. Simulation of multispectral and hyperspectral eo products for onboard machine learning application. *IEEE J. Selected Topics Appl. Earth Observations Remote Sensing* **17**, 17651–17665. https://doi.org/10.1109/JSTARS.2024.3434437 (2024).
13. Rijlaarsdam, D. *et al.* The next era for earth observation spacecraft: An overview of cognisat-6. *IEEE J. Selected Topics Appl. Earth Observations Remote Sensing*. (2024).
14. Marin, A. *et al.* Phi-sat-2: Onboard ai apps for earth observation. *Proc. Space Artif. Intell* **6** (2021).
15. Muñoz, M. M. C. *et al.* Mantis, a 12u smallsat mission taking advantage of super-resolution and artificial intelligence for high-resolution imager. In *Small Satellite Conference* (2024).
16. Choudhary, T., Mishra, V., Goswami, A. & Sarangapani, J. A comprehensive survey on model compression and acceleration. *Artif. Intell. Rev.* **53**, 5113–5155 (2020).
17. Elsken, T., Metzen, J. H. & Hutter, F. Neural architecture search: A survey (2019). arXiv:1808.05377.
18. Poyser, M. & Breckon, T. P. Neural architecture search: A contemporary literature review for computer vision applications. *Pattern Recognit.* **147**, 110052. https://doi.org/10.1016/j.patcog.2023.110052 (2024).
19. Tan, M. & Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, 6105–6114 (PMLR, 2019).
20. Tan, M. *et al.* Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2820–2828 (2019).
21. Su, X. *et al.* Vitas: Vision transformer architecture search. In *European Conference on Computer Vision*, 139–157 (Springer, 2022).
22. Wang, S., Xie, T., Cheng, J., Zhang, X. & Liu, H. Mdl-nas: A joint multi-domain learning framework for vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 20094–20104 (2023).
23. Cai, H., Zhu, L. & Han, S. ProxylessNAS: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations* (2019).
24. Peng, C., Li, Y., Jiao, L. & Shang, R. Efficient convolutional neural architecture search for remote sensing image scene classification. *IEEE Trans. Geosci. Remote Sensing* **59**, 6092–6105 (2020).
25. Li, C. et al. Efficient object detection in sar images based on computation-aware neural architecture search. *Appl. Sci.* **12**, 10978 (2022).
26. Kadway, C. *et al.* Zero-shot embedded neural architecture search for on-board satellite tasks & hardware accelerators. In *Proceedings of SPAICE2024: The First Joint European Space Agency/IAA Conference on AI in and for Space*, 175–179 (2024).
27. Gholami, A. *et al.* A survey of quantization methods for efficient neural network inference. In *Low-power computer vision*, 291–326 (Chapman and Hall/CRC, 2022).
28. He, Y. & Xiao, L. Structured pruning for deep convolutional neural networks: A survey. *IEEE Trans. Pattern Anal. Machine Intell.* **46**, 2900–2919 (2023).
29. Gou, J., Yu, B., Maybank, S. J. & Tao, D. Knowledge distillation: A survey. *Int. J. Computer Vision* **129**, 1789–1819 (2021).
30. Zhang, B., Cheng, D., Zhang, Y., Liu, F. & Tian, J. Lossless model compression via joint low-rank factorization optimization. *arXiv preprint* arXiv:2412.06867 (2024).
31. He, Y., Jiang, L. & Wu, D. Convolutional neural network compression based on low-rank decomposition. *arXiv preprint* arXiv:2408.16289 (2024).
32. Rigamonti, R., Sironi, A., Lepetit, V. & Fua, P. Learning separable filters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2013).
33. Denil, M., Shakibi, B., Dinh, L., Ranzato, M. & de Freitas, N. Predicting parameters in deep learning (2014). arXiv:1306.0543.
34. Papa, L., Russo, P., Amerini, I. & Zhou, L. A survey on efficient vision transformers: Algorithms, techniques, and performance benchmarking. *IEEE Trans. Pattern Anal. Machine Intell.* **46**, 7682–7700. https://doi.org/10.1109/TPAMI.2024.3392941 (2024).
35. Alqahtani, A., Xie, X. & Jones, M. W. Literature review of deep network compression. *Informatics* **8**, 77. https://doi.org/10.3390/informatics8040077 (2021).
36. Rokh, B., Azarpeyvand, A. & Khanteymoori, A. A comprehensive survey on model quantization for deep neural networks in image classification. *ACM Trans. Intell. Syst. Technol.* **14**, 1–50. https://doi.org/10.1145/3623402 (2023).
37. Gong, C. et al. Vecq: Minimal loss dnn model compression with vectorized weight quantization. *IEEE Trans. Comput.* **70**, 696–710. https://doi.org/10.1109/tc.2020.2995593 (2021).
38. Cheng, Y., Wang, D., Zhou, P. & Zhang, T. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Process. Magazine* **35**, 126–136 (2018).
39. Kokhazadeh, M., Keramidas, G., Kelefouras, V. & Stamoulis, I. Denseflex: A low rank factorization methodology for adaptable dense layers in dnns. In *Proceedings of the 21st ACM International Conference on Computing Frontiers*, 21–31 (2024).
40. Wang, L. & Yoon, K.-J. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *IEEE Trans. Pattern Anal. Machine Intell.* **44**, 3048–3068 (2021).
41. Holland, J. H. Genetic algorithms. *Sci. Am.* **267**, 66–73 (1992).
42. Kaelbling, L. P., Littman, M. L. & Moore, A. W. Reinforcement learning: A survey. *J. Artif. Intell. Res.* **4**, 237–285 (1996).
43. Zela, A., Siems, J. & Hutter, F. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. *arXiv preprint* arXiv:2001.10422 (2020).
44. Ru, B., Wan, X., Dong, X. & Osborne, M. Interpretable neural architecture search via bayesian optimisation with weisfeiler-lehman kernels. *arXiv preprint* arXiv:2006.07556 (2020).
45. Lopes, V. F. *Improving Neural Architecture Search With Bayesian Optimization and Generalization Mechanisms*. Ph.D. thesis, Universidade da Beira Interior (Portugal) (2024).
46. Cai, Z., Chen, L., Zeng, S., Lai, Y. & Liu, H.-L. Est-nas: An evolutionary strategy with gradient descent for neural architecture search. *Appl. Soft Comput.* **146**, 110624 (2023).
47. Santra, S., Hsieh, J.-W. & Lin, C.-F. Gradient descent effects on differential neural architecture search: A survey. *IEEE Access* **9**, 89602–89618 (2021).
48. Liu, H., Simonyan, K. & Yang, Y. Darts: Differentiable architecture search. *arXiv preprint* arXiv:1806.09055 (2018).
49. Liu, G., Li, Y., Chen, Y., Shang, R. & Jiao, L. Pol-nas: A neural architecture search method with feature selection for polsar image classification. *IEEE J. Selected Topics Appl. Earth Observations Remote Sensing* **15**, 9339–9354 (2022).
50. Gudzius, P., Kurasova, O., Darulis, V. & Filatovas, E. Automl-based neural architecture search for object recognition in satellite imagery. *Remote Sensing* **15**, 91 (2022).
51. Cassimon, A., Reiter, P., Mercelis, S. & Mets, K. Designing a classifier for active fire detection from multispectral satellite imagery using neural architecture search. *arXiv preprint* arXiv:2410.05425 (2024).
52. Lin, T.-Y., Goyal, P., Girshick, R., He, K. & Dollár, P. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2980–2988, https://doi.org/10.1109/ICCV.2017.324 (2017).
53. NVIDIA Corporation. Nvidia jetson agx orin series (2022). Accessed: 2025-05-02.
54. Intel Corporation. Intel® movidius™ myriad ™ x vision processing unit (2017). Accessed: 2025-05-02.

55. Pultarova, T. Spacex to launch 1st space-hardened nvidia ai gpu on upcoming rideshare mission (2024). Accessed: 2025-05-02.
56. Ubotica Technologies. D-orbit's wild ride ion mission (2021). Accessed: 2025-05-02.
57. Vasu, P. K. A., Gabriel, J., Zhu, J., Tuzel, O. & Ranjan, A. Mobileone: An improved one millisecond mobile backbone. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 7907–7917 (2023).
58. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778 (2016).
59. European Space Agency (ESA). The discovery campaign on cognitive cloud computing (2025). Accessed on 2025-09-03.
60. Meoni, G. *et al.* E2e: Onboard satellite real-time classification of thermal hotspots events on optical raw data. *Astrodynamics* 1–17 (2025).

Acknowledgements
We extend our sincere gratitude to Prof. Michał Przewoźniczek (Associate Professor, Department of Computational Intelligence, Faculty of Computer Science and Management, Wroclaw University of Science and Technology, Wroclaw, Poland), Prof. Jakub Nalepa (Associate Professor, Department of Algorithmics and Software, Faculty of Automatic Control, Electronics and Computer Science, Silesian University of Technology, Gliwice, Poland), and Prof. Jan Van Rijn (Leiden Institute of Advanced Computer Science, Leiden University, The Netherlands) for their invaluable guidance and insightful feedback.

Author contributions
R.D.P. conceived the experiment(s), P.K.T. and M.W. conducted the experiment(s), R.D.P. and L.P., and P.K.T. analysed the results. All authors reviewed the manuscript.

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-025-21467-8.

Correspondence and requests for materials should be addressed to R.D.P.