# Deployment-Aligned Low-Precision Neural Architecture Search for Spaceborne Edge AI

Anonymous CVPR submission

Paper ID 25

## Abstract

*Designing deep networks that meet strict latency and accuracy constraints on edge accelerators increasingly relies on hardware-aware optimization, including neural architecture search (NAS) guided by device-level metrics. Yet most hardware-aware NAS pipelines still optimize architectures under full-precision assumptions and apply low-precision adaptation only after the search, leading to a mismatch between optimization-time behavior and deployment-time execution on low-precision hardware that can substantially degrade accuracy. We address this limitation by integrating deployment-aligned low-precision training directly into the hardware-aware NAS loop. Candidate architectures are exposed to FP16 numerical constraints during fine-tuning and evaluation, enabling joint optimization of architectural efficiency and numerical robustness without modifying the search space or evolutionary strategy. We evaluate the proposed framework on vessel segmentation for spaceborne maritime monitoring, targeting the Intel® Movidius™ Myriad™ X Vision Processing Unit (VPU) accelerator. While post-training precision conversion reduces on-device performance from 0.85 to 0.78 mIoU, deployment-aligned low-precision training achieves 0.826 mIoU on-device for the same architecture (95,791 parameters), recovering approximately two-thirds of the deployment-induced accuracy gap without increasing model complexity. These results demonstrate that incorporating deployment-consistent numerical constraints into hardware-aware NAS substantially improves robustness and alignment between optimization and deployment for resource-constrained edge (Artificial Intelligence) AI systems.*

## 1. Introduction

Neural Architecture Search (NAS) has emerged as a powerful paradigm for automatically designing deep neural networks that balance accuracy and efficiency under resource constraints [1]. Hardware-aware NAS extends this idea by incorporating device-level performance metrics such as latency, throughput, or memory footprint directly into the optimization loop, enabling architectures to be selected with deployment constraints in mind [16]. However, despite explicitly accounting for hardware characteristics, hardware-aware NAS pipelines still optimize candidate architectures under full-precision floating-point (FP32) training assumptions and apply low-precision adaptation only after the search is complete [14]. This decoupling introduces a systematic mismatch between optimization-time behavior and deployment-time execution on low-precision edge accelerators, often resulting in substantial accuracy degradation once models are deployed.

In this work, we address this limitation by integrating deployment-aligned low-precision effects directly into the hardware-aware NAS evaluation loop. Rather than treating reduced numerical precision as a post-training concern, candidate architectures are exposed to low-precision floating point (FP16) numerical constraints during fine-tuning and on-device evaluation. This allows the NAS process to jointly optimize architectural efficiency and numerical robustness without modifying the search space or the underlying evolutionary strategy.

This problem is particularly relevant in Earth Observation (EO), where missions increasingly explore on-board data processing to enable faster and more autonomous decision-making in spaceborne systems [25, 27]. Applications such as maritime monitoring [28], disaster response [2, 15, 23], and environmental surveillance [31] benefit from reducing the latency between image acquisition and event detection. Traditional EO pipelines rely on downlinking raw or minimally processed imagery to ground stations, introducing delays due to limited visibility windows, bandwidth constraints, and the need for multiple ground station passes before actionable information can be generated [33].

Recent missions and demonstrators have shown that performing inference directly on-board can significantly reduce these delays by transmitting extracted information instead of raw data [24]. However, this shift toward on-board intelligence places stringent constraints on the computa-

tional resources available for data processing [17]. Spaceborne platforms particularly small satellites and CubeSatsoperate under tight limits on power consumption, memory footprint, and processing throughput, requiring deep learning models that are both compact and efficient [13].

Designing neural networks that satisfy these constraints while maintaining task performance remains challenging. Simply reducing model size or complexity does not reliably preserve accuracy, especially for dense prediction tasks such as semantic segmentation [5]. Moreover, space missions increasingly rely on heterogeneous commercial off-the-shelf edge accelerators, which exhibit distinct architectural and numerical characteristics despite operating within similar power envelopes [17]. As a result, models trained and optimized under conventional full-precision settings are particularly vulnerable to deployment-time performance degradation when executed under reduced numerical precision.

Hardware-aware NAS has been successfully applied to spaceborne inference, demonstrating favorable accuracy–efficiency trade-offs on edge platforms through device-in-the-loop evaluation [14]. Nevertheless, existing approaches typically rely on post-training precision conversion to adapt selected architectures to low-precision hardware, leaving numerical robustness largely unaddressed during the search itself. This limitation motivates the need for NAS pipelines that explicitly align optimization-time assumptions with deployment-time numerical behavior.

We evaluate the proposed deployment-aligned training strategy on a vessel segmentation task for maritime monitoring, using the same dataset and experimental protocol as recent NAS-based EO studies [30]. All candidate architectures are evaluated directly on an Intel® Movidius™ Myriad™ X Vision Processing Unit(VPU) accelerator, enabling a realistic assessment of segmentation accuracy, inference latency, and throughput under operational edge deployment conditions. The results show that incorporating deployment-consistent numerical constraints within the NAS loop substantially reduces the performance gap between Graphics Processing Unit (GPU)-based optimization and on-device inference, while maintaining compact model size and real-time execution capability.

The main contributions of this work are summarized as follows:

- **Deployment-aligned NAS:** We integrate low-precision-aware training directly into a hardware-aware NAS loop, aligning optimization-time behavior with deployment-time FP16 execution without modifying the search space or evolutionary strategy.
- **Search-level numerical alignment:** We show that incorporating deployment-consistent precision during evaluation changes architecture selection dynamics and reduces the optimization, deployment accuracy gap compared to post-training precision conversion.
- **Realistic device-in-the-loop validation:** We provide comprehensive on-device evaluation on an Intel® Movidius™ Myriad™ X VPU, measuring segmentation accuracy and throughput under real deployment conditions.

The remainder of this paper is organized as follows. Section 2 reviews related work on model compression, neural architecture search, and low-precision deployment. Section 3 presents the proposed hardware-aware NAS framework with deployment-aligned low-precision training. Section 4 reports quantitative and qualitative experimental results. Section 5 concludes the paper and discusses future research directions.

## 2. Background and Related Work

### 2.1. On-Board Earth Observation and Edge Constraints

Deploying deep learning models on-board spacecraft requires aggressive reductions in model size and computational footprint due to strict constraints on memory capacity, non-volatile storage, power consumption, and thermal dissipation imposed by on-board computers and edge accelerators [11]. Space-qualified hardware platforms typically provide limited on-chip memory and restricted external storage, making it infeasible to deploy large, high-capacity neural networks without careful optimization.

These constraints are particularly relevant for EO missions, where on-board processing is increasingly adopted to reduce downlink latency and enable timely decision-making. However, the combination of dense prediction tasks, such as semantic segmentation, and heterogeneous commercial off-the-shelf edge accelerators poses significant challenges for reliable deployment. As a result, model efficiency and deployment robustness have become central concerns in spaceborne AI pipelines.

### 2.2. Hardware-Aware Neural Architecture Search

Rather than compressing an already trained model, NAS aims to discover efficient neural network architectures from scratch by formulating architecture design as a constrained optimization problem over a predefined search space $\mathcal{A}$. The objective is to identify an architecture that minimizes a task-specific loss while satisfying deployment-related constraints such as latency, memory footprint, or computational cost:

$$\min_{a \in \mathcal{A}} \mathcal{L}(a, \mathcal{D}) \quad \text{s.t.} \quad C(a) \leq C_{\max}. \tag{1}$$

Hardware-aware NAS extends this paradigm by incorporating device-level performance metrics directly into the optimization loop, enabling architectures to be selected with deployment constraints in mind [16]. Early NAS approaches relied on population-based or reinforcement learning strategies [20, 29], while more recent methods improve

efficiency through weight sharing, differentiable relaxation, or direct device-aware optimization [6, 22, 32].

NAS has been successfully applied to remote sensing and EO tasks, including object recognition, classification, and onboard inference [9, 18, 21]. In particular, prior work has demonstrated the feasibility of embedding hardware evaluation within the NAS loop for spaceborne deployment, achieving favorable accuracy–efficiency trade-offs on edge accelerators [14]. However, most hardware-aware NAS pipelines optimize candidate architectures under full-precision training assumptions and adapt them to low-precision hardware only after the search is complete.

### 2.3. Low-Precision Deployment and Quantization-Aware Training

To enable efficient execution on resource-constrained hardware, model compression techniques such as quantization, pruning, knowledge distillation, and low-rank decomposition are commonly employed [11]. Among these, quantization is particularly effective at reducing memory footprint, computational cost, and energy consumption by lowering the numerical precision of weights and activations [26].

Post-training quantization (PTQ) applies precision reduction after full-precision training, typically relying on calibration data to estimate quantization parameters [4, 7]. While PTQ enables rapid deployment, it is sensitive to numerical mismatch and often leads to accuracy degradation on compact architectures and low-bit precision hardware [12].

Quantization-aware training (QAT) mitigates this issue by explicitly modeling low-precision effects during training using fake-quantization operators and straight-through estimators [26]. QAT generally yields higher robustness at deployment time, but it is typically applied after architecture selection and treated as a separate post-optimization step [8, 10].

### 2.4. Limitations of Prior Work

Despite advances in both hardware-aware NAS and quantization-aware training, these two lines of research have largely evolved independently. Existing approaches either (i) incorporate hardware metrics into NAS while optimizing architectures under full-precision assumptions, or (ii) apply low-precision or quantization-aware training only after the architecture has been selected. This separation introduces a mismatch between optimization-time behavior and deployment-time execution, particularly for spaceborne edge accelerators operating under reduced numerical precision.

This limitation motivates the integration of deployment-aware numerical constraints directly into the NAS process. By exposing candidate architectures to low-precision effects during evaluation, the search itself can favor architectures that are not only structurally efficient, but also numerically robust under real deployment conditions.

## 3. Methodology

This work extends hardware-aware NAS by explicitly aligning the numerical conditions observed during optimization with those encountered at deployment time on edge accelerators. While existing hardware-aware NAS frameworks incorporate device-level performance metrics (e.g., latency or throughput) through post-training on-device evaluation, they typically rely on full-precision (FP32) training on GPUs and implicitly assume numerical equivalence at deployment. In practice, edge accelerators frequently execute reduced-precision arithmetic, introducing discrepancies in dynamic range, rounding behavior, and accumulation error that are not captured during FP32 optimization.

Our key idea is to integrate deployment-aligned low-precision effects directly into the NAS evaluation loop. Instead of treating precision conversion as a post-selection adaptation step, candidate architectures are exposed to the same numerical regime used at deployment during fine-tuning and on-device evaluation. This biases the search toward architectures that are not only structurally efficient, but also numerically robust under the target execution precision.

Figure 1 summarizes the proposed workflow. For each candidate architecture sampled by the genetic algorithm, we evaluate two deployment scenarios: (i) **post-training precision conversion (PTQ baseline)**, where an FP32-trained model is exported to an FP16 OpenVINO intermediate representation (IR) and executed on the target device, and (ii) **deployment-aligned low-precision training**, where the same architecture is fine-tuned under FP16-aware projections prior to export and on-device evaluation. Importantly, both branches share the same architecture, search space, and evolutionary operators; only the evaluation pathway differs.

### 3.1. Hardware-Aware Neural Architecture Search

**Search space:** The admissible set of candidate architectures is defined as a discrete search space $\mathcal{A}$ composed of single-path networks with up to $M_n$=6 learnable blocks. Each block is sampled from a library of convolutional primitives and macro-blocks: (i) convolutional layers with activation (`ConvAct`) or with batch normalization (`ConvBnAct`), (ii) squeeze-and-excitation convolutional blocks (`ConvSE`), (iii) mobile inverted bottleneck blocks with and without residual connections (`MBConv`, `MBConvNoRes`), (iv) cross-stage partial variants (`CSPConvBlock`, `CSPMBConvBlock`), (v) DenseNet-style blocks (`DenseNetBlock`), (vi) ResNet-style blocks (`ResNetBlock`), with optional pooling (`AvgPool`, `MaxPool`), dropout, and a fixed segmentation head. Within each primitive, discrete and continuous

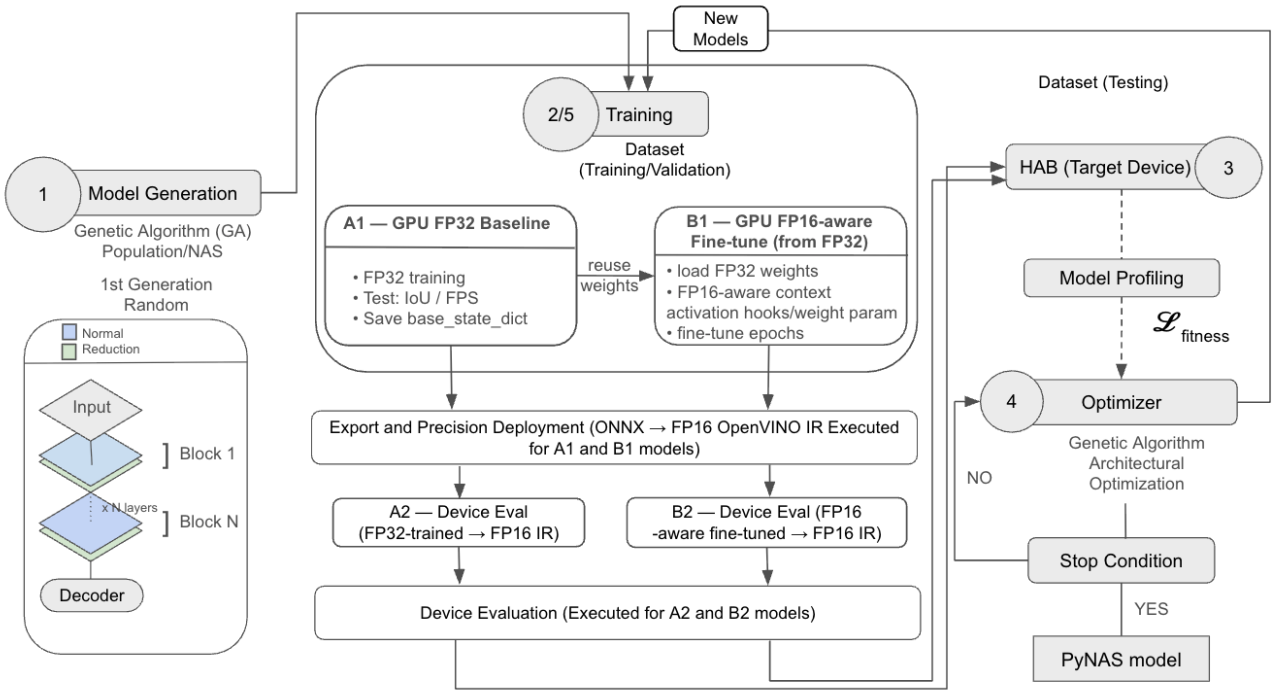


Figure 1. The diagram illustrates the proposed NAS framework in which each candidate architecture generated by the genetic algorithm is trained and evaluated under two deployment-aligned numerical regimes. First, an FP32 baseline model is trained on GPU (A1) and directly exported to FP16 OpenVINO intermediate representation, enabling post-training precision conversion (PTQ-style deployment) and subsequent on-device evaluation (A2). In parallel, the same architecture is fine-tuned using low-precision-aware training (B1), initialized from the FP32 weights and incorporating fake-FP16 activation rounding and weight parametrization with straight-through estimators, before being exported to FP16 OpenVINO IR and evaluated on the target hardware (B2). Both FP32-trained and FP16-aware fine-tuned models are therefore deployed and benchmarked on the target device, allowing a direct comparison between PTQ-based deployment and deployment-aligned low-precision training. The NAS fitness $\mathcal{L}$ is computed from on-device performance metrics, while the paired evaluation paths enable quantitative and qualitative assessment of accuracy degradation due to post-training conversion and the corresponding recovery achieved through low-precision-aware fine-tuning, as reported in the results.

hyperparameters are sampled from predefined ranges, including kernel size (1–5 for standard conv blocks), channel-width coefficients (e.g., 4–24 depending on block type), and activation functions (ReLU or GELU). An architecture is represented by a structured *architecture code* (a compact genotype string) that deterministically parses into a valid network, enabling mutation and crossover while preserving structural validity.

**Genetic algorithm settings:** We use a population-based evolutionary NAS strategy with population size $P_s$=16 and $G$=10 generations. At each generation, candidates are ranked using a hardware-aware fitness function and updated through (i) elitism, retaining the top $k_{\text{best}}$=1 architectures, (ii) diversity injection, sampling $n_{\text{random}}$=6 new architectures uniformly from $\mathcal{A}$, and (iii) genetic operators applied to the mating pool consisting of the top 50% of the population. Mutation is applied with probability $p_{\text{mut}}$=0.15, and single-point crossover recombines parent genotypes to generate offspring.

**Fitness:** The fitness of an architecture $a$ is computed as

$$\mathcal{F}(a) = \alpha \cdot \text{fps}(a) + \beta \cdot \text{Metric}(a) \cdot e^{\gamma \cdot \text{Metric}(a)}, \quad (2)$$

where Frames Per Second $\text{fps}(a)$ is inference throughput measured on the target device and $\text{Metric}(a)$ is task accuracy (mIoU). The exponential term increases sensitivity to high-accuracy solutions, biasing the search toward architectures that satisfy both predictive and deployment constraints.

**Device-in-the-loop evaluation.** After training (and optional fine-tuning, described below), each candidate architecture is exported to an FP16 OpenVINO intermediate representation (IR) and executed on the Intel® Movidius™ Myriad™ X VPU. Throughput and latency are measured directly on-device.

This captures hardware-specific effects such as operator scheduling, memory access patterns, and reduced-precision arithmetic behavior that are not observable during GPU training. These measurements feed back into the NAS loop via $\mathcal{F}(a)$.

### 3.2. Deployment-Aligned Low-Precision Training

**Training budget per candidate:** Each candidate architecture is trained for $E_{\text{FP32}}{=}10$ epochs in FP32 using batch size 32. To model deployment-time reduced precision, we then perform FP16-aware fine-tuning for $E_{\text{LP}}{=}10$ epochs (after a 1-epoch warmup), initialized from the FP32-trained weights. This staged procedure isolates the effect of deployment-aligned numerical adaptation: the PTQ baseline and the FP16-aware model share the same initialization and architecture.

**FP16 projection operator and affected layers:** Rather than integer-only quantization, we model reduced-precision effects through FP16 projection consistent with the target deployment pipeline. Let $f_\theta$ denote a network with parameters $\theta$. During fine-tuning, we inject a projection operator $\mathcal{P}(\cdot)$ into the forward pass:

$$y = \mathcal{P}\big(f_\theta(x)\big), \tag{3}$$

where $\mathcal{P}$ casts activations to FP16 and back to FP32 to simulate reduced mantissa precision and rounding. Projections are applied to the outputs of convolutional and linear operators (i.e., the primary accumulation sites), while gradients are accumulated in FP32. Weight rounding is enabled during the forward pass (`fp16_round_weights=true`) using a straight-through estimator (STE),

$$\frac{\partial \mathcal{P}(x)}{\partial x} \approx 1, \tag{4}$$

so training remains stable while the forward computation reflects reduced precision.

**Activation clipping.** To better match the dynamic range observed under FP16 deployment and reduce overflow sensitivity, activations are clipped during FP16-aware fine-tuning using a fixed symmetric bound $|x| \leq 12$. This empirically improves numerical stability and reduces sensitivity to rounding-induced amplification during inference.

**FP16 projection for Myriad X deployment:** Our deployment pipeline exports networks to an FP16 OpenVINO intermediate representation (IR) and executes them on the Intel® Movidius™ Myriad™ X VPU. FP16-aware projections therefore emulate the dominant numerical effect introduced by the deployment toolchain reduced mantissa precision and rounding in FP16 arithmetic while maintaining compatibility with standard FP32 optimizers. Although the Intel® Movidius™ Myriad™ X VPU may still exhibit additional device-specific behavior (e.g., operator fusion or internal accumulation details), using FP16 projections during fine-tuning directly targets the same precision regime used at inference and substantially reduces the optimization–deployment mismatch observed with post-training conversion.

**Integration into NAS:** For each candidate architecture, we first compute an on-device PTQ baseline by exporting the FP32-trained model to FP16 IR and evaluating it on the target device. We then fine-tune the same architecture under FP16-aware projections, re-export, and re-evaluate on-device. Fitness is computed from the deployment-aligned (fine-tuned) on-device metrics, guiding evolution toward architectures that are both hardware-efficient and numerically robust.

## 4. Results

### 4.1. Dataset

All experiments are conducted using the HRSC2016 dataset, a publicly available benchmark for high-resolution ship detection and segmentation in optical satellite imagery. The dataset consists of images primarily collected from Google Earth, with ground sampling distances ranging from approximately 0.4 m to 2 m. Image dimensions vary significantly, spanning from roughly $300 \times 300$ pixels to over $1500 \times 900$ pixels, with most samples exceeding $1000 \times 600$ pixels in size.

Original annotations are provided as oriented bounding boxes with associated vessel categories, covering both civilian and military ships across 25 classes. In this work, the task is formulated as binary vessel segmentation, where all ship categories are merged into a single foreground class and the remaining pixels are treated as background. This formulation reflects realistic onboard processing requirements, where timely and reliable vessel localization is often prioritized over fine-grained classification.

To obtain pixel-level supervision, bounding box annotations were converted into dense segmentation masks. Images were intensity-normalized and converted to a common format, after which annotation files were parsed to extract bounding box information. These bounding boxes were used to generate initial segmentation masks, which were subsequently refined through additional processing and manual inspection to ensure accurate vessel coverage. All samples were resized to $512 \times 512$ pixels using zero-padding when necessary and stored as NumPy arrays together with their corresponding masks for efficient training and evaluation.

Additional preprocessing steps were applied to improve

dataset quality and robustness. In particular, targeted augmentation was used to address imbalances between small and large vessels, while corrupted or low-quality samples were removed. Extreme or ambiguous annotations were manually reviewed. This preprocessing pipeline results in a clean and diverse dataset suitable for evaluating both segmentation accuracy and deployment robustness under reduced numerical precision.

### 4.2. Evaluation Metrics

Segmentation performance is evaluated using the mean Intersection-over-Union (mIoU), computed over all valid classes and averaged across the batch. The mIoU computation follows the same rules used during on-device evaluation, excluding classes with empty unions and ignoring invalid pixels when applicable, ensuring consistency between GPU-based evaluation and deployment-time measurements.

Architecture selection during the NAS process is driven by the hardware-aware fitness function defined in Eq. (3), which jointly accounts for segmentation accuracy and inference throughput measured directly on the target device.

### 4.3. Experimental Setup

All candidate architectures are trained on a workstation equipped with an AMD Radeon™ GPU using full-precision (FP32) arithmetic [3]. Deployment-time evaluation is performed on the Intel® Movidius™ Myriad™ X vision processing unit, a low-power edge accelerator representative of onboard computing platforms used in spaceborne systems [19].

Two experimental configurations are evaluated within the same hardware-aware NAS framework:

- **Post-training precision conversion (baseline):** Architectures are trained exclusively in FP32 on the GPU. After training, models are exported and evaluated on the Intel® Movidius™ Myriad™ X VPU using post-training precision conversion, without exposure to reduced-precision effects during optimization.
- **Low-precision-aware fine-tuning:** Architectures are first trained in FP32 and subsequently fine-tuned with low-precision-aware training, where FP16 numerical effects are explicitly modeled during the forward pass. The fine-tuned models are exported and evaluated on the Intel® Movidius™ Myriad™ X VPU, and the resulting on-device metrics are used directly in the NAS fitness loop.

Both configurations share the same search space, evolutionary strategy, dataset, and evaluation protocol, enabling a direct comparison of the impact of low-precision-aware training. This setup closely follows prior hardware-aware NAS experiments, with the key distinction that numerical precision effects are now incorporated during optimization rather than deferred to deployment.

### 4.4. Results

Figure 2 summarizes the quantitative behavior of the hardware-aware NAS process under post-training precision conversion and low-precision-aware fine-tuning. The figure reports IoU–FPS trade-offs, fitness evolution across generations, and the progression of maximum accuracy and throughput for both configurations.

Under post-training precision conversion, a pronounced discrepancy is observed between GPU-based evaluation and on-device inference. As shown in Fig. 2(a), many architectures that achieve high IoU under full-precision GPU evaluation experience a substantial degradation when deployed on the Intel® Movidius™ Myriad™ X. This effect is particularly evident for architectures with higher parameter counts, indicating sensitivity to deployment-time numerical constraints. While inference throughput remains competitive, the resulting accuracy drop limits the effectiveness of these models for onboard segmentation.

The evolution of fitness values across generations further highlights this limitation. As shown in Fig. 2(a), fitness improvements plateau early, and the gap between maximum and median fitness remains relatively large, indicating unstable convergence and a limited number of architectures that jointly satisfy accuracy and efficiency constraints after deployment.

In contrast, low-precision-aware fine-tuning leads to a markedly different optimization behavior. As shown in Fig. 2(b), architectures optimized under low-precision awareness achieve consistently higher on-device IoU for comparable inference throughput. The IoU–FPS distribution shifts upward, indicating improved robustness to deployment-time numerical effects. Notably, the spread of solutions narrows across generations, suggesting that a larger fraction of the population remains viable under reduced precision.

Fitness trends further confirm this behavior. The maximum fitness increases steadily across generations, while the fitness gap between elite and median architectures decreases, indicating improved population-wide robustness. Importantly, the best-performing architecture achieves a GPU mIoU of approximately 0.84 while maintaining an on-device mIoU close to 0.82 on the Intel® Movidius™ Myriad™ X VPU, significantly reducing the accuracy gap observed under post-training precision conversion.

In addition to quantitative improvements, qualitative inspection reveals clear differences in deployment robustness across numerical configurations. Figure 3 compares segmentation outputs produced under GPU FP32 execution, post-training precision conversion on the Intel® Movidius™ Myriad™ X VPU, and deployment-aligned FP16 training. While GPU FP32 predictions closely match the ground truth masks, post-training precision conversion frequently introduces structural artifacts during on-device ex-

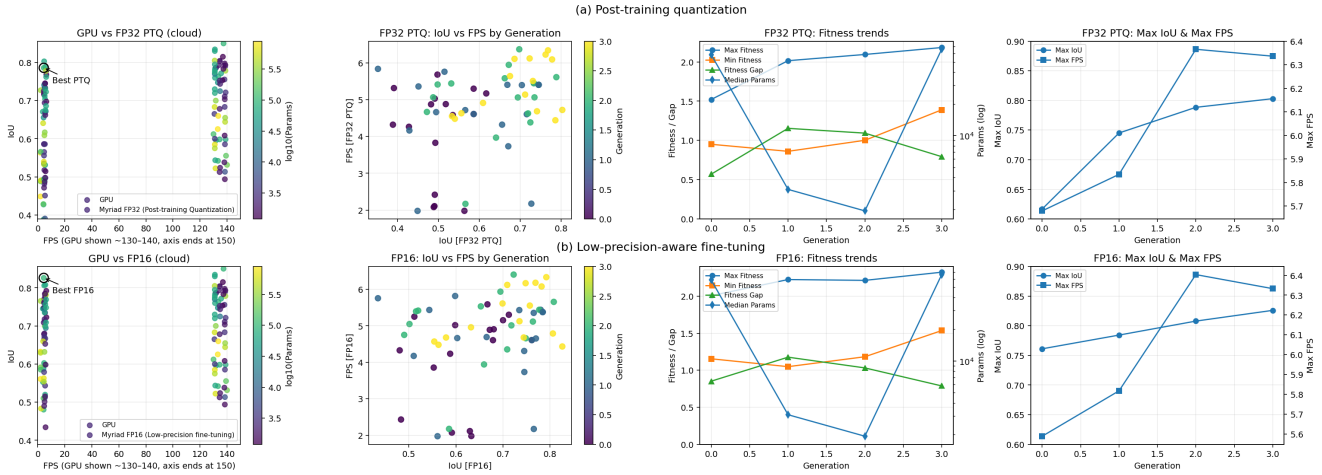


Figure 2. Quantitative comparison between post-training precision conversion and low-precision-aware fine-tuning within the hardware-aware NAS framework. Top row: PTQ-based optimization. Bottom row: low-precision-aware fine-tuning. From left to right: IoU–FPS trade-offs comparing GPU and Myriad VPU execution, IoU versus FPS evolution across generations, fitness trends over generations, and maximum IoU and FPS progression. Low-precision-aware training consistently reduces deployment-induced accuracy loss and improves convergence stability on the target device.

ecution, including fragmented vessel contours, small holes within predicted regions, and increased false negatives for elongated or low-contrast targets. These errors are consistent with the numerical sensitivity observed in the quantitative results and reflect the mismatch between optimization-time and deployment-time behavior.

In contrast, models optimized with deployment-aligned low-precision training preserve vessel morphology more reliably under on-device inference. FP16-aware fine-tuning reduces contour fragmentation, maintains connectivity along vessel hulls, and improves recall for small or partially occluded targets. Notably, these qualitative gains are achieved without increasing model complexity, confirming that numerical robustness emerges from exposure to deployment-consistent precision during optimization rather than from architectural overparameterization.

Overall, both quantitative metrics and qualitative analysis demonstrate that incorporating low-precision effects directly into the NAS optimization loop substantially improves the alignment between optimization-time behavior and deployment-time performance. By reducing deployment-induced accuracy loss and preserving segmentation structure under on-device execution without modifying the search space or evolutionary strategy the proposed approach yields architectures that are both efficient and reliable for real-time onboard vessel segmentation.

## 5. Conclusion

This work extends hardware-aware NAS by explicitly aligning optimization-time training with the numerical conditions encountered at deployment on low-precision edge accelerators. By integrating deployment-aligned FP16-aware training directly into the NAS evaluation loop, we reduce the mismatch between GPU-based optimization and on-device execution without modifying the search space or evolutionary strategy.

Experiments on vessel segmentation demonstrate that exposure to deployment-consistent numerical effects during optimization substantially improves on-device robustness. On the Intel® Movidius™ Myriad™ X VPU, low-precision–aware optimization recovers a large fraction of the accuracy lost under post-training precision conversion, while preserving compact model size and real-time inference performance.

Beyond the specific hardware platform studied, these results highlight a broader methodological insight: architectural efficiency alone is insufficient for reliable edge deployment if numerical behavior is ignored during optimization. For spaceborne and other safety-critical edge AI systems, training pipelines must reflect deployment-time precision constraints to avoid brittle performance in operation.

Future work will explore extending this approach to integer-only and mixed-precision accelerators, as well as jointly optimizing numerical robustness, energy efficiency, and latency within a unified hardware-aware NAS framework. Ultimately, by unifying architecture search, hardware feedback, and numerical-awareness, this work contributes toward a more deployment-centric view of neural optimization, one that is essential for reliable, autonomous intelligence at the edge of space.

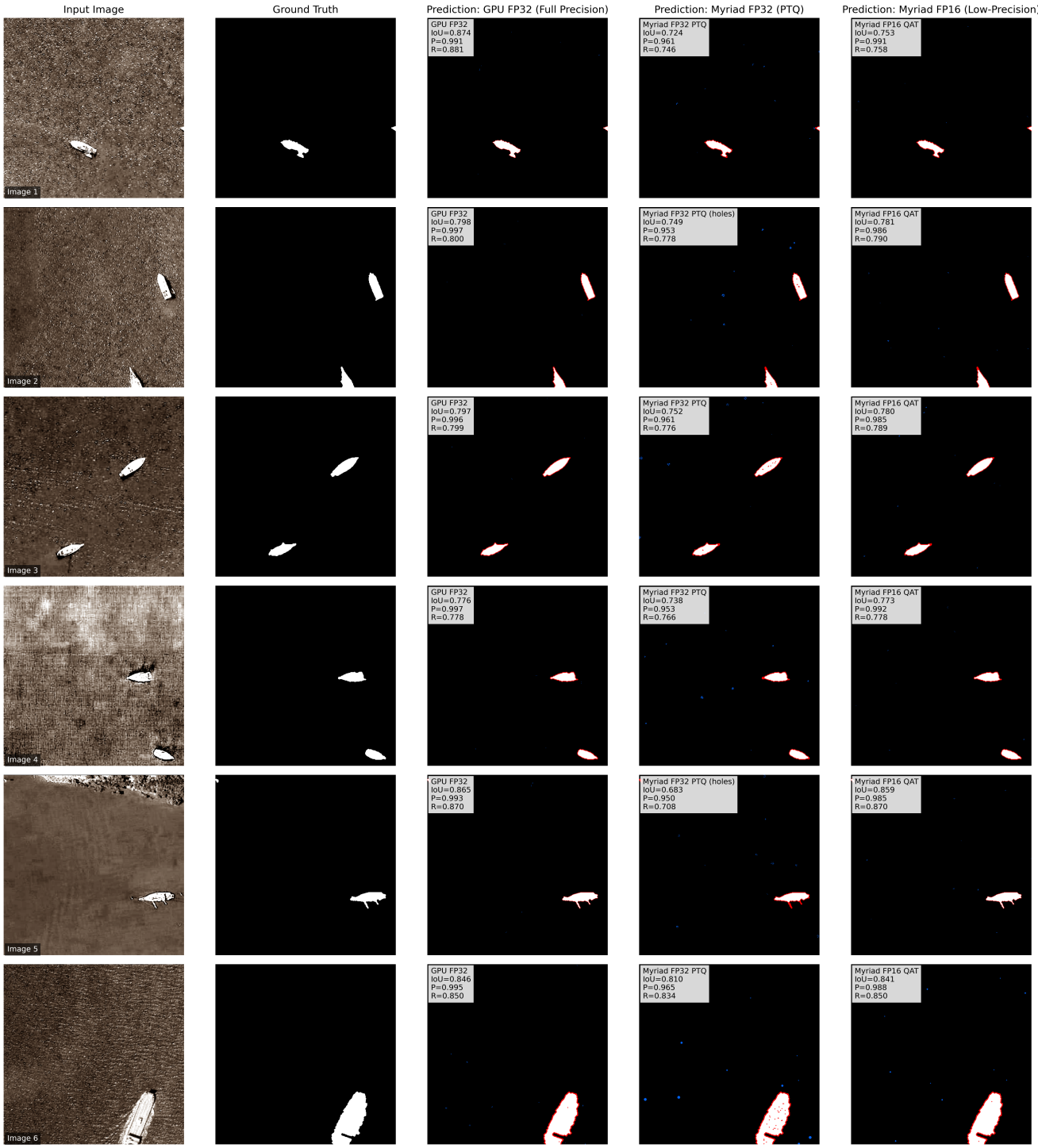


Figure 3. Qualitative comparison of vessel segmentation under different numerical conditions. From left to right: input image, ground truth mask, GPU FP32 prediction, Myriad X VPU prediction after post-training precision conversion, and Myriad X VPU prediction after deployment-aligned FP16 training. False negatives (missed vessel pixels) are highlighted in red, while false positives (spurious detections) are highlighted in blue. Post-training precision conversion introduces fragmented contours, internal holes, and missed detections, particularly for elongated or low-contrast vessels. In contrast, deployment-aligned low-precision training preserves vessel connectivity and morphology more reliably under on-device execution.

# References


[1] Neural architecture search: A contemporary literature review for computer vision applications. *Pattern Recognition*, 147: 110052, 2024. 1

[2] Towards onboard thermal hotspots segmentation with raw multispectral satellite imagery. *International Journal of Applied Earth Observation and Geoinformation*, 146:105095, 2026. 1

[3] Advanced Micro Devices, Inc. AMD Radeon™ Graphics. 6

[4] Ron Banner, Yury Nahshan, Elad Hoffer, and Daniel Soudry. Post-training 4-bit quantization of convolution networks for rapid-deployment, 2019. 3

[5] Ayoub Benali Amjoud and Mustapha Amrouch. Object detection using deep learning, CNNs and vision transformers: A review. *IEEE Access*, PP:1–1, 2023. 2

[6] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware, 2019. 3

[7] Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. Zeroq: A novel zero shot quantization framework, 2020. 3

[8] Léopold Cambier, Anahita Bhiwandiwalla, Ting Gong, Mehran Nekuii, Oguz H Elibol, and Hanlin Tang. Shifted and squeezed 8-bit floating point format for low-precision training of deep neural networks, 2020. 3

[9] Amber Cassimon, Phil Reiter, Siegfried Mercelis, and Kevin Mets. Designing a classifier for active fire detection from multispectral satellite imagery using neural architecture search, 2024. 3

[10] Brian Chmiel, Liad Ben-Uri, Moran Shkolnik, Elad Hoffer, Ron Banner, and Daniel Soudry. Neural gradients are near-lognormal: improved quantized and sparse training, 2020. 3

[11] Tejalal Choudhary, Vipul Mishra, Anurag Goswami, and Jagannathan Sarangapani. A comprehensive survey on model compression and acceleration. *Artificial Intelligence Review*, 53(7):5113–5155, 2020. 2, 3

[12] Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. Low-bit quantization of neural networks for efficient inference, 2019. 3

[13] Angela Cratere, Leandro Gagliardi, Gabriel A. Sanca, Federico Golmar, and Francesco Dell'Olio. On-board computer for CubeSats: State-of-the-art and future trends. *IEEE Access*, 12:99537–99569, 2024. 2

[14] Roberto Del Prete, Parampuneet Kaur Thind, Andrea Mazzeo, Matthew Whitley, Lorenzo Papa, Nicolas Longépé, and Gabriele Meoni. Optimizing deep learning models for on-orbit deployment through neural architecture search. *Scientific Reports*, 15(1):37783, 2025. 1, 2, 3

[15] Maria Pia Del Rosso, Alessandro Sebastianelli, Dario Spiller, Pierre Philippe Mathieu, and Silvia Liberata Ullo. On-board volcanic eruption detection through CNNss and satellite multispectral imagery. *Remote Sensing*, 13(17): 3479, 2021. 1

[16] Jin-Dong Dong, An-Chieh Cheng, Da-Cheng Juan, Wei Wei, and Min Sun. Dpp-net: Device-aware progressive search for pareto-optimal neural architectures, 2018. 1, 2

[17] Gianluca Furano, Gabriele Meoni, Aubrey Dunne, David Moloney, Veronique Ferlet-Cavrois, Antonis Tavoularis, Jonathan Byrne, Léonie Buckley, Mihalis Psarakis, Kay-Obbe Voss, and Luca Fanucci. Towards the use of artificial intelligence on the edge in space systems: Challenges and opportunities. *IEEE Aerospace and Electronic Systems Magazine*, 35(12):44–56, 2020. 2

[18] Povilas Gudzius, Olga Kurasova, Vytenis Darulis, and Ernestas Filatovas. AutoML-based neural architecture search for object recognition in satellite imagery. *Remote Sensing*, 15(1), 2023. 3

[19] Intel Corporation. Intel® Movidius™ Myriad™ X Vision Processing Unit. 6

[20] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996. 2

[21] Guangyuan Liu, Yangyang Li, Yanqiao Chen, Ronghua Shang, and Licheng Jiao. Pol-nas: A neural architecture search method with feature selection for polsar image classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 15:9339–9354, 2022. 3

[22] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search, 2019. 3

[23] Gonzalo Mateo-Garcia, Joshua Veitch-Michaelis, Lewis Smith, Silviu Vlad Oprea, Guy Schumann, Yarin Gal, Atılım Güneş Baydin, and Dietmar Backes. Towards global flood mapping onboard low cost satellites with machine learning. *Scientific Reports*, 11(1):7249, 2021. 1

[24] Nicola Melega, Bernardo A. Carnicero, Nicolas Longépé, Aiste Paskeviciute, Valerio Marchese, Oriol Aragon, Irina Babkina, Alessandro Marin, Jakub Nalepa, and Leonie Buckley. Implementation of the $\phi$sat-2 on-board image processing chain. In *Sensors, Systems, and Next-Generation Satellites XXVII*, pages 264–276. SPIE, 2023. 1

[25] Nicola Melega, Nicolas Longepe, Agne Paskeviciute, Valentina Marchese, Oriol Aragon, Irina Babkina, Alessandro Marin, Jakub Nalepa, Leonie Buckley, Giorgia Guerrisi, Sofia Oliveira, and Hano Steyn. Development and implementation of the ΦSat-2 mission. In *Small Satellites Systems and Services Symposium (4S 2024)*, page 135462X. International Society for Optics and Photonics, SPIE, 2025. 1

[26] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization, 2021. 3

[27] Massimiliano Pastena, Michel Tossaint, Amanda Regan, Michele Castorina, Pierre Mathieu, Josep Rosello, Antonio Gabriele, and Nicola Melega. Overview of ESA's Earth Observation upcoming small satellites missions. 2020. 1

[28] Roberto Del Prete, Gabriele Meoni, Nicolas Longépé, Maria Daniela Graziano, and Alfredo Renga. First results of vessel detection with onboard processing of sentinel-2 raw data by deep learning. In *IGARSS 2023 - 2023 IEEE International Geoscience and Remote Sensing Symposium*, pages 6262–6265, 2023. 1

[29] Colin Reeves. *Genetic Algorithms*, pages 109–139. 2010. 2

[30] Parampuneet Thind, Roberto Prete, Andrea Mazzeo, Nicolas Longépé, and Gabriele Meoni. Naseo: Neural architecture search for Earth Observation onboard processing. 2026. 2

[31] UN-SPIDER. AI-enabled onboard edge computing for satellite intelligence in disaster management, 2025. 1

[32] Arber Zela, Julien Siems, and Frank Hutter. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search, 2020. 3

[33] Bing Zhang, Yuanfeng Wu, Boya Zhao, Jocelyn Chanussot, Danfeng Hong, Jing Yao, and Lianru Gao. Progress and challenges in intelligent remote sensing satellite systems. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 15:1814–1822, 2022. 1