ENAS: An Efficient Hardware-Aware Neural Architecture Search Framework for TinyML on Resource-Constrained Microcontrollers

Mohd Moin Khan[1,*], Naman Srivastava[1] and Pandarasamy Arjunan[1]

[1]*Indian Institute of Science (IISc), Bengaluru 560012, India*

Abstract

We present **ENAS**, a hardware-aware Neural Architecture Search (NAS) framework that combines a static feasibility check, a cell-based search space supporting standard, depthwise-separable, and bottleneck blocks with optional skip connections, and a three-stage hybrid search strategy (random $\rightarrow$ top-K $\rightarrow$ mutation) with persistent cross-run caching. Unlike many existing NAS frameworks that rely on GPU acceleration, ENAS is designed to operate efficiently without requiring GPUs, making it suitable for resource-constrained development environments. We evaluate ENAS on two TinyML benchmarks, Visual Wake Words and Melanoma Cancer, across eight microcontrollers with memory footprints ranging from 20 KB to 1 MB SRAM and nine input image resolutions. Our experimental results show that ENAS achieves mean search-time speedups of $2.41\times$ and $1.70\times$ on the Visual Wake Words and Melanoma Cancer datasets, respectively, while maintaining competitive test accuracy compared with the recent NanoNAS framework. A measured resource analysis further shows that ENAS-selected models use substantially lower peak activation RAM, the binding constraint for microcontroller deployment at matched accuracy. Additionally, ENAS achieves 79.4% test accuracy on an STM32H743-based microcontroller, outperforming the greedy CPU-only baseline by 2.6 percentage points. We release the ENAS framework as open-source at: https://github.com/EdgeIntelligenceLab/ENAS

Keywords

Hardware-aware Neural Architecture Search, TinyML, Microcontrollers, CPU-only NAS, Edge AI, Quantization-Aware Deployment

1. Introduction

TinyML enables the deployment of deep learning models on microcontrollers (MCUs) operating under kilobyte-scale SRAM and milliwatt-scale power budgets, enabling always-on perception for battery-powered IoT devices. Designing efficient TinyML models are largely driven by hardware-aware Neural Architecture Search (NAS), where candidate models must satisfy strict deployment constraints, including peak activation RAM, Flash storage for weights, and multiply-accumulate (MACC) budgets that approximate inference latency and energy consumption. The MACC budgets used in this paper follow the convention established by NanoNAS [1]: they are application-driven latency targets rather than the hardware's raw capability, and are intentionally set below the peak device capability.

Recent TinyML NAS frameworks such as **MCUNet** [2] and **MicroNAS** [3] achieve strong accuracy, but require substantial GPU resources for search and training (e.g., MCUNet reports approximately $\sim$300 GPU hours). Such requirements limit their accessibility for many edge AI researchers, embedded developers, and small IoT laboratories. More recent CPU-only NAS approaches, such as **NanoNAS** [1], a hardware-aware NAS framework that has been shown to outperform several existing TinyML NAS methods, improve accessibility by avoiding GPU dependence. However, NanoNAS relies on a greedy search strategy over a relatively constrained search space. It also incurs significant overhead from repeated TFLite conversion and full model retraining during candidate evaluation. Furthermore, its search space cannot effectively represent architectural primitives such as depthwise-separable convolutions, bottleneck blocks, and skip connections, which are known to be critical for efficient TinyML

SuRE'26: 1st Workshop on Sustainability and Resource-Efficiency of Artificial Intelligence, co-located with IJCAI 2026, August 17, 2026, Bremen, Germany

*Corresponding author.

moinkhanmohd@iisc.ac.in (M. M. Khan); snaman@iisc.ac.in (N. Srivastava); samy@iisc.ac.in (P. Arjunan)

deployments [4, 5].

To address these limitations, this paper presents **ENAS**, a CPU-only hardware-aware NAS framework for efficient TinyML model development targeting computer vision applications on resource-constrained microcontrollers. Built on top of the NanoNAS framework, ENAS extends its hardware-aware search pipeline with lightweight analytical feasibility estimation, a richer cell-based search space, and a more efficient hybrid search strategy. ENAS replaces NanoNAS's per-candidate *measured* feasibility check (which converts each candidate to TFLite-Micro and measures RAM/Flash before deciding feasibility) with a lightweight static *analytical* estimator for RAM, Flash, and MACC constraints (Eq. 3), eliminating most TFLite conversions during the search. It further introduces a flexible cell-based search space supporting standard, depthwise-separable, and bottleneck blocks with optional skip connections. Instead of greedy coordinate ascent, ENAS employs a three-stage hybrid strategy consisting of random sampling, top-K candidate selection, and local mutation, together with persistent cross-run caching to reduce redundant evaluations.

Unlike prior work that typically evaluates NAS performance on a limited set of hardware platforms or input configurations, we conduct a comprehensive study across two TinyML applications (Visual Wake Words and Melanoma Cancer classification), eight MCU platforms, and nine input image resolutions. This results in 636 fully trained and deployed models, enabling a detailed analysis of deployment-level behavior across diverse hardware-resource regimes, including feasibility boundaries, search-time scaling, and the resource footprint of selected architectures, and the impact of richer search spaces on constrained MCUs.

Contributions.

1. We propose **ENAS**, a CPU-only hardware-aware NAS framework built on top of NanoNAS, combining a flexible cell-based search space and a three-stage hybrid search algorithm for TinyML deployment.
2. We design analytical RAM, Flash, and MACC static feasibility checks that eliminate expensive late-stage TFLite conversion during candidate evaluation, reducing per-candidate evaluation time from 5–10 minutes to approximately $\sim$30 seconds.
3. We conduct a large-scale evaluation across 8 MCU platforms, 9 input resolutions, and 2 TinyML datasets, resulting in 636 fully trained models with zero deployment pipeline failures, and report mean$\pm$standard-deviation statistics together with Wilcoxon significance tests throughout.
4. ENAS achieves mean search-time speedups of $2.41\times$ on Visual Wake Words and $1.70\times$ on Melanoma Cancer compared with the greedy CPU-only NanoNAS baseline (both highly significant, $p < 10^{-8}$), while maintaining competitive accuracy with only small mean accuracy reductions of 0.79 pp and 1.31 pp, respectively.
5. Using measured (on-tool) RAM and Flash footprints, we show that ENAS-selected architectures use markedly lower peak activation RAM ($\sim 0.38\times$ at matched accuracy on both datasets), the resource that binds first on MCUs, at the cost of higher Flash and MACC, and we identify deployment regimes where richer search spaces provide accuracy benefits (up to $+10.6$ pp on STM32H743) as well as scenarios where the simpler greedy baseline remains preferable.

2. Related Works

Neural architecture search. The NAS literature spans reinforcement-learning controllers [6], differentiable methods with Gumbel-softmax relaxations [7, 8], evolutionary search, and weight-sharing supernets [9]. These methods deliver state-of-the-art accuracy at mobile latency budgets but require hundreds to tens of thousands of GPU hours. Path binarisation [8] and one-shot supernet training [9] reduce this cost but still target mobile and server hardware rather than kilobyte-scale microcontrollers. Zero-cost proxies [10] eliminate training during search, though their rank correlation with final accuracy degrades on sub-100 KB architectures.

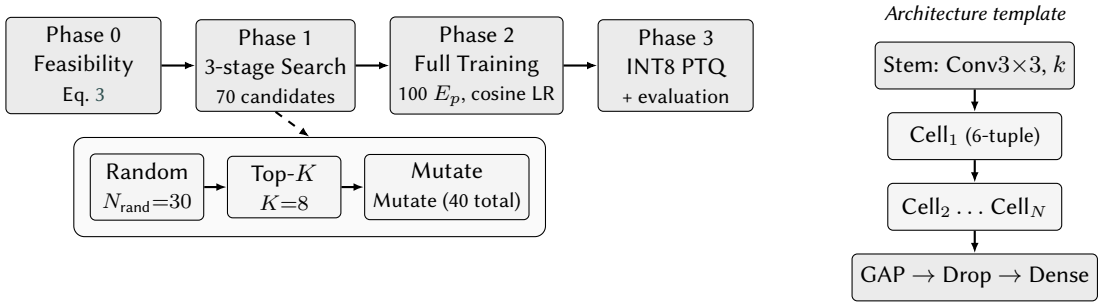

Figure 1: ENAS pipeline (left) and cell-based architecture template (right). Phase 0 eliminates infeasible candidates analytically. Phase 1 evaluates 70 candidates (30 random + 40 mutants (5 per top-K survivor)) using short proxy training. Each cell is independently configured across six dimensions: block type, kernel size, stride, skip connection, activation, and expansion ratio.

NAS for Microcontrollers. Microcontroller-targeted NAS introduces rigid SRAM and Flash constraints that compress the search space and necessitate tight co-design with downstream inference runtimes. Within this domain, MCUNet [2] couples a two-stage TinyNAS search with the customized TinyEngine runtime, achieving over 70% ImageNet top-1 accuracy on commercial MCUs; subsequent extensions integrate patch-based inference [11] and enable on-device training within a tight 256 KB budget [12]. Similarly, MicroNAS [3] leverages differentiable NAS alongside latency lookup tables for time-series classification on MCUs. To minimize optimization overhead, zero-shot MCU variants [13, 14] achieve substantial search speedups via training-free proxies, while the recent ELASTIC [15] framework expands once-for-all supernet search to object detection on microcontrollers.

Crucially, all of these frameworks depend on GPU compute or intensive supernet pre-training. To eliminate this bottleneck, NanoNAS [1] descended from ColabNAS [16] employs a CPU-only, greedy coordinate-ascent search over filter count k and depth c, while TinyTNAS [17] introduces a parallel CPU-only, time-bounded approach for time-series workloads. The proposed ENAS framework belongs to this sustainable, CPU-only family; however, unlike ELASTIC and related supernet methodologies, it operates entirely without GPU acceleration or pre-training overhead, and unlike NanoNAS, it navigates a significantly richer, cell-based search space.

Limitations of existing CPU-only NAS. Three limitations of the greedy CPU-only baseline emerge from measured experiments.
(L1) Search-time bottleneck: for every candidate, NanoNAS performs a *measured* feasibility check: it converts the candidate to TFLite-Micro and measures peak RAM/Flash only *after* the candidate has been trained, so the search phase consumes 60–76 % of total runtime (9–155 min per configuration). ENAS *replaces this measured check with an analytical one* performed before training.
(L2) Narrow search space: the (k, c) parameterisation excludes depthwise-separable convolution, bottlenecks, skip connections, and stride control. These primitives are parameter- and activation-efficient; depthwise-separable and bottleneck blocks reduce per-layer multiply-accumulate cost relative to standard convolution [18, 5].
(L3) Greedy instability: per-board accuracy varies by up to 13 pp between runs on the same configuration on STM32H743 at 96×96, indicating local optima in coordinate ascent.

3. The ENAS Framework

Hardware-aware Neural Architecture Search (HW-NAS) for TinyML can be formulated as a constrained optimisation problem. Given a search space of candidate architectures $\mathcal{S}$, the goal is to identify an architecture $\mathcal{A}^\star \in \mathcal{S}$ that maximises deployment accuracy while satisfying the hardware constraints of a target microcontroller:

$$\mathcal{A}^\star = \arg\max_{\mathcal{A}\in\mathcal{S}} \mathrm{Acc}(\mathcal{A}) \tag{1}$$

subject to:

$$R(\mathcal{A}) \leq R_{\max}, \quad F(\mathcal{A}) \leq F_{\max}, \quad M(\mathcal{A}) \leq M_{\max}, \tag{2}$$

where $R(\mathcal{A})$, $F(\mathcal{A})$, and $M(\mathcal{A})$ denote the peak SRAM usage, Flash footprint, and multiply-accumulate (MACC) operations of architecture $\mathcal{A}$, and $(R_{\max}, F_{\max}, M_{\max})$ are the hardware budgets.

NanoNAS adopts a CPU-only greedy search and evaluates each candidate through repeated TFLite conversion, proxy training, and a measured hardware feasibility validation. While this reduces dependence on GPU clusters, the search remains expensive due to repeated post-hoc measurement and retraining, and its restricted (k, c) space cannot capture efficient primitives such as depthwise-separable convolutions, bottleneck layers, and skip connections.

To address these limitations, ENAS extends NanoNAS with three improvements: **(1)** static analytical feasibility estimation to eliminate unnecessary TFLite conversion, **(2)** a richer cell-based search space, and **(3)** a three-stage hybrid search strategy with persistent cross-run caching. ENAS is a four-phase pipeline (Fig. 1): **(0)** static feasibility check, **(1)** cell-based search, **(2)** full training, and **(3)** INT8 post-training quantisation (PTQ) and evaluation. The framework is designed for efficient operation on a single CPU node without GPU acceleration.

3.1. Static Feasibility Check

Given hardware constraints $(R_{\max}, F_{\max}, M_{\max})$ and input shape (H, W, C_{in}), ENAS computes a lower bound on peak activation memory:

$$R_{\min} = H \cdot W \cdot 4 + H \cdot W \cdot C_{\text{in}} \quad \text{bytes}. \tag{3}$$

Equation (3) is *not* a per-layer estimate: it is an input-dependent lower bound on the network's peak activation footprint, combining the largest single activation tensor that any feasible network must materialise (the $H \cdot W \cdot 4$ term, a working buffer) with the input tensor itself ($H \cdot W \cdot C_{\text{in}}$). Because every architecture in $\mathcal{S}$ must at least hold the input and one activation tensor, any configuration with $R_{\min} > R_{\max}$ is provably infeasible and is rejected before training. The check is therefore a necessary (not sufficient) condition and assumes no relation such as $C_{\text{in}} = C_{\text{out}}$ in any layer; intermediate-layer feasibility is verified analytically per candidate during search (Algorithm 1, line 4).

Across the 72 hardware-resolution configurations evaluated per dataset, the analytical check identified all infeasible cases in advance. As shown in Fig. 2, infeasibility follows the predictable boundary $R_{\min} \propto H \cdot W \cdot k \cdot 4$ bytes, matching the exact points where NanoNAS later fails during its measured post-hoc TFLite deployment check.

3.2. Cell-Based Search Space

Each architecture is represented as $\mathcal{A} = (k, [\mathbf{c}_1, \ldots, \mathbf{c}_n])$, where $k \in \{1, 2, \ldots, 16\}$ denotes the number of stem convolution filters and $n \in \{1, \ldots, 6\}$ the number of cells. Each cell $\mathbf{c}_i$ is parameterised as $\mathbf{c}_i = (b, \kappa, s, g, a, e)$, where $b \in \{\text{standard, depthwise-separable, bottleneck}\}$ is the block type, $\kappa \in \{3, 5\}$ the kernel size, $s \in \{1, 2\}$ the stride, $g \in \{\text{true, false}\}$ the skip-connection flag, $a \in \{\texttt{relu}, \texttt{relu6}\}$ the activation, and $e \in \{1, 2\}$ the expansion ratio. During Stage 1, random sampling draws the stem width from the constrained range $k \leq 12$ to bias exploration toward compact stems on tight budgets; Stage 3 mutation samples k from the full $\{1, \ldots, 16\}$ range, so the realised search space spans $k \in \{1, \ldots, 16\}$ (selected values in our experiments range from 1 to 16). Per-cell channel counts are *not* searched independently: they are derived deterministically from the stem width k via a fixed expansion schedule $n_{\text{out}} = \lceil n \cdot m_i \rceil$ with a decaying multiplier m_i, mirroring NanoNAS's channel-doubling rule. This keeps the search space tractable ($\sim 10^4$ structural configurations after fixing the channel schedule) while letting the cell tuple control the qualitatively important primitives. The cell options are intentionally restricted to the primitives with the highest accuracy-per-byte return on MCUs (small kernels, binary stride, low expansion); broadening them is straightforward but enlarges the search budget (Section 6). Skip connections improve gradient flow when tensor dimensions match, while `relu6` improves compatibility with INT8 PTQ due to its bounded activation range.

Input	L010 RBT6	NUCLEO L010	Nano 33IoT	NUCLEO L412	RPi Pico	Nano 33BLE	Nicla Vision	STM32 H743
32×32	✓	✓	✓	✓	✓	✓	✓	✓
48×48	✓	✓	✓	✓	✓	✓	✓	✓
50×50	✓	✓	✓	✓	✓	✓	✓	✓
64×64	✗	✗	✓	✓	✓	✓	✓	✓
72×72	✗	✗	✗	✓	✓	✓	✓	✓
80×80	✗	✗	✗	✓	✓	✓	✓	✓
96×96	✗	✗	✗	✓	✓	✓	✓	✓
112×112	✗	✗	✗	✗	✓	✓	✓	✓
128×128	✗	✗	✗	✗	✓	✓	✓	✓
Feasible	3/9	3/9	4/9	7/9	9/9	9/9	9/9	9/9

Figure 2: Analytical activation-memory boundary. Peak RAM scales with $H \cdot W \cdot k \cdot 4$ bytes, accurately predicting the infeasible regions later detected by NanoNAS during its measured TFLite deployment check.

3.3. Three-Stage Hybrid Search

Algorithm 1 presents the overall search strategy for ENAS. It approximates the optimal architecture through a three-stage hybrid strategy. ENAS first evaluates N_{rand}=30 randomly sampled architectures, selects the top K=8 candidates, and generates N_{mut}=5 mutations per survivor, producing 40 mutated candidates in total. The budget $(N_{\text{rand}}, K, N_{\text{mut}}) = (30, 8, 5)$ yields a 70-candidate pass that balances feasible-region coverage and search cost on resource-constrained MCUs: $N_{\text{rand}} = 30$ ensures sufficient feasible exploration, $K = 8$ focuses mutation on high-performing candidates, and $N_{\text{mut}} = 5$ provides near-complete coverage of the single-edit neighbourhood of the 6-dimensional cell space. To avoid redundant evaluations, ENAS maintains a persistent cross-run cache of previously evaluated architectures for the same hardware and input configuration.

Each candidate is scored using:

$$s = w_{\text{acc}}\, v_{\text{acc}} + w_{\text{eff}}\left(1 - \sqrt[3]{\frac{r}{R_{\max}}\frac{f}{F_{\max}}\frac{m}{M_{\max}}}\right) + w_{\text{hr}}\, h, \tag{4}$$

where $h = \mathbf{1}\left[\max\left(\frac{r}{R_{\max}}, \frac{f}{F_{\max}}, \frac{m}{M_{\max}}\right) \le 0.8\right]$. The three terms encode complementary objectives. v_{acc} is the proxy validation accuracy after E_p=3 epochs on 30 % of the training data, and w_{acc} weights it. w_{eff} weights an *efficiency* term: one minus the geometric mean of the three normalised resource ratios, which rewards architectures that leave headroom on all three budgets simultaneously (the geometric mean prevents a single slack constraint from masking a tight one). w_{hr} weights a binary *headroom* indicator h, which fires only when *every* resource ratio is ≤ 0.8; it is a discrete bonus, deliberately separate from the continuous efficiency term, that protects a margin for INT8 PTQ overhead and fine-tuning that the smooth efficiency term alone does not guarantee at the 0.8 boundary. We set $(w_{\text{acc}}, w_{\text{eff}}, w_{\text{hr}}) = (0.80, 0.15, 0.05)$.

Stage 2 performs a two-phase ranking. First, candidates are filtered by proxy validation accuracy alone; the remaining top 50 % are then ranked using the full scoring function in Eq. 4, and the top-K architectures are retained for mutation. When a previously feasible architecture exists for the same hardware platform, ENAS may inject it as a *warm seed* (refer to Section 4.3) into the Stage 1 candidate pool.

3.4. Calibration of Proxy Training and Scoring

The effectiveness of ENAS depends on the proxy training budget E_p and the weight w_{acc}. An initial configuration using E_p=1 and w_{acc}=0.65 produced a Spearman rank correlation of ρ=0.18 (n=30, not significant) between proxy validation accuracy and final TFLite INT8 test accuracy, too low for reliable ranking. The richer cell-based space introduces skip-connection paths and deeper architectures whose optimisation dynamics require more than a single proxy epoch to become informative. Increasing E_p to 3 improves the rank correlation to ρ=0.71 (p<0.001). Increasing w_{acc} to 0.80 further prevents the

Algorithm 1 ENAS Hybrid Search

Input: hardware constraints $(R_{\max}, F_{\max}, M_{\max})$, input shape $\mathbf{x}$, proxy epochs E_p, scoring weights $(w_{\text{acc}}, w_{\text{eff}}, w_{\text{hr}})$, search budget $(N_{\text{rand}}, K, N_{\text{mut}})$, optional warm-seed architecture $\mathcal{A}_0$.
Output: best architecture $\mathcal{A}^\star$

1: Compute $R_{\min}$ via Eq. (3); if $R_{\min} > R_{\max}$, **return infeasible**.
2: Load persistent cache $\mathcal{C}$.
3: *Stage 1 (Random).* Sample N_{rand} random architectures; if $\mathcal{A}_0$ is available, include it in the pool.
4: In parallel, for each $\mathcal{A}$: query $\mathcal{C}$; else estimate (r, f, m) analytically. If any constraint is violated, mark infeasible; otherwise proxy-train E_p epochs on 30 % data and score by Eq. (4).
5: *Stage 2 (Top-K).* Apply the accuracy-first filter, then the full score of Eq. (4), and retain the top K.
6: *Stage 3 (Mutation).* Generate N_{mut} mutated architectures per survivor and evaluate them in parallel using Eq. (4).
7: **return** $\mathcal{A}^\star = \arg\max s(\mathcal{A})$ over Stages 1 and 3.

Table 1
Target MCU platforms.

Platform	RAM	Flash	MACC
STM32L010RBT6	20 KB	128 KB	0.75 M
NUCLEO-L010RB	20 KB	64 KB	0.75 M
Arduino Nano33IoT	32 KB	256 KB	1.20 M
NUCLEO-L412KB	64 KB	128 KB	3.20 M
Raspberry Pi Pico	264 KB	2 MB	3.00 M
Arduino Nano33BLE	256 KB	1 MB	4.00 M
Arduino Nicla Vision	1 MB	2 MB	8.00 M
STM32H743ZI	1 MB	2 MB	15.0 M

efficiency term from dominating selection when proxy accuracies fall within a narrow range. All ENAS results use the calibrated setting $(E_p{=}3, w_{\text{acc}}{=}0.80)$ unless otherwise specified.

4. Experimental Setup

4.1. Datasets

We evaluate on two TinyML benchmarks chosen for their distinct difficulty regimes: **Visual Wake Words** [19], binary person-presence on natural images (a vision benchmark of moderate difficulty, $\sim$73% baseline accuracy), and **Melanoma Cancer**, a medical-imaging benchmark with binary classification of dermoscopic images and stronger class signal ($\sim$89% baseline accuracy). Together they test whether ENAS generalises across signal regimes.

4.2. Hardware Platforms

The evaluation considers eight distinct MCU platforms (Table 1) spanning three resource tiers: ultra-constrained (with 20 KB of SRAM), mid-tier, and high-capacity (up to 1 MB of SRAM). Across these platforms, the target hardware configuration budgets range from 0.75 M to 15.00 M MACC operations.

4.3. Protocol

We evaluate square input resolutions $P_X \times P_X$, with $P_X \in \{32, 48, 50, 64, 72, 80, 96, 112, 128\}$. Across the resulting $8 \times 9 = 72$ (hardware, resolution) configurations, 53 are feasible under Eq. (3) and 19 are correctly rejected. We run each feasible cell three times per method per dataset, totalling $53 \times 3 \times 2 \times 2 = 636$ fully trained models. Full training uses 100 epochs with cosine LR decay

Table 2
Aggregate results across all 53 feasible (hardware, resolution) cells per dataset. Each cell statistic is the mean over 3 runs; aggregate values are mean±std over the 53 cell means. p-values are from two-sided Wilcoxon signed-rank tests comparing ENAS and NanoNAS for each dataset.

Metric	VWW		Cancer	
	NanoNAS	**ENAS**	**NanoNAS**	**ENAS**
Mean acc. (%)	72.68 ± 2.58	71.89 ± 3.33	89.44 ± 1.26	88.13 ± 1.52
Median acc. (%)	72.71	72.24	89.77	88.23
Mean search (min)	102.2 ± 44.6	42.4 ± 21.2	10.3 ± 3.8	6.1 ± 2.6
Speedup	**2.41×**		**1.70×**	
Acc. p-value	0.118 (n.s.)		3.4×10^{-6}	
Search p-value	4.7×10^{-10}		5.9×10^{-9}	

($10^{-2} \rightarrow 10^{-6}$), batch size 128, and `EarlyStopping` (patience 15). INT8 PTQ uses 150 representative samples; peak RAM and Flash of the resulting model are measured with STMicroelectronics' `stm32tflm` tool, identical to NanoNAS, so all measured resource numbers are directly comparable.

Warm seeding. A *warm seed* is an architecture from a prior, already-completed search on the *same* hardware/resolution encoded as a standard-convolution cell sequence with the NanoNAS-best (k, c), that is prepended to the Stage 1 candidate pool. It guarantees a non-empty feasible pool on tight RAM budgets where random sampling may produce no feasible cell, and anchors top-K selection with a known-good accuracy floor. Unless stated otherwise, the broad cross-resolution sweep (Tables 2, 5, 6, 3) is run *without* warm seeding; the focused $50\times50/64\times64$ study (Table 4) uses warm seeding, which is why its selected architectures differ. We compare ENAS against NanoNAS [1] on identical hardware budgets and report an intermediate ablation (*ENAS-strategy*) that retains the 2-D search space with parallel random search to isolate strategy from search space.

5. Results

5.1. Aggregate Trade-off

Table 2 summarizes the aggregate performance metrics. ENAS achieves a mean search-time speedup of $2.41\times$ on VWW and $1.70\times$ on Cancer, accompanied by minor average accuracy reductions of 0.79 pp and 1.31 pp, respectively. Statistical analysis reveals a clear distinction between these two outcomes: the search-time reduction is highly significant across both datasets ($p < 10^{-8}$), whereas the accuracy difference is not statistically significant on VWW ($p = 0.118$) and remains small on Cancer despite reaching significance. Standard deviations are slightly higher under ENAS, a consequence of the larger search space exposing more architectural variation, but median accuracy gaps remain tight (0.47 pp and 1.54 pp). Search time is the discriminative dimension: ENAS strictly dominates on per-configuration runtime.

5.2. Resource Footprint of Selected Models

Table 3 evaluates whether richer architectural primitives yield superior efficiency in deployed models. The results reveal a nuanced trade-off that is highly advantageous for TinyML microcontrollers. Specifically, ENAS-selected architectures require significantly less peak activation RAM than NanoNAS: approximately $0.30\times$ on VWW and $0.21\times$ on Cancer across all cells, and a consistent $0.38\times$ on both datasets within the matched-accuracy subset. Conversely, ENAS models utilize more Flash (2.85–$5.73\times$ at matched accuracy) and higher MACC (1.39–$3.94\times$). This trade-off aligns precisely with real-world microcontroller constraints, where peak SRAM serves as the primary binding bottleneck; indeed, every infeasible cell encountered in this study was strictly RAM-infeasible, making the activation-memory

Table 3
Measured resource footprint of selected models, mean±std over the 53 feasible cells (RAM and Flash measured with `stm32tflm`; MACC analytical). The final two rows give the ENAS/NanoNAS ratio over all cells and restricted to matched-accuracy cells ($|\Delta\text{acc}| \leq 1$ pp).

Dataset	Method	RAM (KB)	Flash (KB)	MACC (M)
VWW	NanoNAS	40.4 ± 28.6	14.0 ± 4.6	1.77 ± 1.49
	ENAS	12.1 ± 5.5	53.8 ± 37.8	2.56 ± 2.44
Cancer	NanoNAS	37.0 ± 27.3	9.4 ± 2.4	0.86 ± 0.86
	ENAS	7.6 ± 4.1	50.7 ± 42.0	1.85 ± 1.72
Ratio (all, VWW / Cancer)		0.30 / 0.21	3.84 / 5.40	1.44 / 2.15
Ratio (matched acc.)		0.38 / 0.38	2.85 / 5.73	1.39 / 3.94

Table 4
TFLite INT8 test accuracy (%, mean±std over 3 runs) at the two most commonly-targeted TinyML input sizes on VWW, using warm seeding (see Section 4.3). ENAS surpasses NanoNAS on the mean at 64×64 and on three of six platforms.

Input 50×50

Hardware	NanoNAS	ENAS	Δ
L010RBT6	72.7 ± 1.4	69.9 ± 3.1	-2.9
NUCLEO-L010	72.0 ± 1.3	68.1 ± 3.8	-3.9
Nano33IoT	71.8 ± 1.9	70.4 ± 1.2	-1.4
NUCLEO-L412	72.0 ± 2.7	71.2 ± 2.3	-0.8
RPi Pico	73.5 ± 1.1	72.0 ± 1.0	-1.5
Nano33BLE	73.7 ± 0.2	73.0 ± 1.4	-0.7
Nicla Vision	74.6 ± 0.2	74.1 ± 3.1	-0.5
STM32H743	70.6 ± 6.1	75.1 ± 2.5	$+4.5$
Mean	72.6	71.7	-0.9

Input 64×64

Hardware	NanoNAS	ENAS	Δ
Nano33IoT	71.2 ± 3.2	72.0 ± 1.1	$+0.8$
NUCLEO-L412	74.0 ± 1.8	72.0 ± 1.7	-2.0
RPi Pico	72.9 ± 3.4	74.4 ± 1.3	$+1.4$
Nano33BLE	71.2 ± 4.1	73.8 ± 1.8	$+2.7$
Nicla Vision	73.2 ± 2.4	75.9 ± 2.0	$+2.6$
STM32H743	74.7 ± 1.6	74.7 ± 3.1	$+0.0$
Mean	73.0	73.7	$+0.7$

bound in Eq. (3) the critical limiting factor. In contrast, Flash and computational capacity are comparatively abundant on these platforms, as evidenced by the fact that all 636 selected models easily fit within the physical hardware budgets. The depthwise-separable and bottleneck blocks, together with stride-based downsampling and skip connections, let ENAS produce architectures with smaller peak activation tensors at comparable accuracy. This demonstrates that the cell-based framework does not simply offer an alternative search space, but actively biases optimization toward RAM-efficient topologies. Finally, while the analytical Flash term in Eq. (4) serves as a loose lower bound for early filtering, the empirical `stm32tflm` measurements confirm that physical feasibility holds in practice because the conservative RAM estimate remains the definitive pre-flight constraint.

5.3. Focused Study at 50×50 and 64×64 (VWW)

Across the two most frequently targeted input dimensions (Table 4), ENAS achieves competitive mean accuracy, yielding a minor deficit of -0.90 pp at a 50×50 resolution and a net gain of $+0.70$ pp at 64×64. Notably, at the 64×64 input resolution, ENAS outperforms the baseline alternative on four of the six feasible hardware platforms, which includes an accuracy improvement of $+2.60$ pp on the Nicla Vision and $+2.70$ pp on the Nano33BLE.

5.4. Cross-Resolution Sweep

Table 5 consolidates the per-resolution accuracy and search-time performance metrics across both datasets. Two distinct behavioral patterns emerge from these data. First, small-input regimes systematically favor ENAS; for instance, at a 32×32 resolution on VWW, ENAS yields an accuracy improvement

Table 5
Per-resolution accuracy (%, mean±std over the N feasible cells) and search time (min) across all feasible hardware platforms. Δ = ENAS − NanoNAS in percentage points; speedup = NanoNAS/ENAS. The mean speedups ($2.41\times$ on VWW, $1.70\times$ on Cancer) hold uniformly across resolutions, while accuracy varies by tier and input size.

		Visual Wake Words						**Melanoma Cancer**					
		Accuracy (%)			*Search (min)*			*Accuracy (%)*			*Search (min)*		
Input	N	**NanoNAS**	**ENAS**	Δ	**Nano**	**ENAS**	**Sp.**	**NanoNAS**	**ENAS**	Δ	**Nano**	**ENAS**	**Sp.**
32^2	8	69.6 ± 3.1	71.6 ± 1.9	$+2.05$	58	39	1.50	89.9 ± 1.1	89.2 ± 1.3	-0.69	9	6	1.49
48^2	8	71.4 ± 1.2	72.1 ± 4.3	$+0.66$	82	37	2.25	90.0 ± 1.0	88.4 ± 1.5	-1.57	10	5	1.82
50^2	8	72.6 ± 1.3	70.9 ± 3.8	-1.77	103	37	2.80	89.6 ± 1.0	88.4 ± 1.0	-1.22	10	5	1.90
64^2	6	72.9 ± 1.4	72.6 ± 3.4	-0.24	108	41	2.62	89.1 ± 1.8	88.5 ± 1.1	-0.58	10	6	1.64
72^2	5	74.0 ± 0.8	72.7 ± 3.2	-1.29	120	47	2.53	88.4 ± 1.7	88.3 ± 1.4	-0.12	8	6	1.29
80^2	5	72.4 ± 2.8	72.9 ± 4.3	$+0.52$	91	47	1.92	89.4 ± 1.1	88.5 ± 0.6	-0.83	12	7	1.73
96^2	5	75.7 ± 1.2	72.1 ± 3.4	-3.62	150	48	3.11	89.9 ± 0.7	86.4 ± 0.1	-3.51	13	7	1.97
112^2	4	75.0 ± 1.6	70.7 ± 4.4	-4.27	144	51	2.84	89.3 ± 1.5	87.3 ± 2.0	-2.02	13	7	1.87
128^2	4	73.8 ± 3.2	71.6 ± 2.5	-2.21	111	47	2.36	88.4 ± 1.0	86.5 ± 2.4	-1.93	10	6	1.61
Mean	53	72.7 ± 2.6	71.9 ± 3.3	-0.79	102	42	**2.41**	89.4 ± 1.3	88.1 ± 1.5	-1.31	10	6	**1.70**

of $+2.05$ percentage points (pp). In this low-resolution regime, the richer architectural primitives particularly depthwise-separable cells utilizing a stride of 2 effectively compensate for constrained spatial information. Second, large-input regimes conversely favor NanoNAS, with ENAS exhibiting an accuracy deficit of 2.00–4.30 pp across the 96×96–112×112 resolution range on both datasets.

This performance degradation stems from a proxy-ranking limitation rather than an inherent deficiency in the search space. At larger input resolutions, the expanded feasible space accommodates numerous candidate architectures whose 3-epoch proxy validation accuracies are statistically indistinguishable, yet their fully converged final accuracies diverge substantially. Consequently, the brief proxy evaluation phase fails to reliably rank the upper tail of the architecture distribution, a phenomenon directly reflected in the elevated standard deviation of ENAS at these resolutions. The same mechanism explains the consistent 50×50 shortfall: 50×50 sits just above the feasibility boundary for the two 20 KB devices, leaving a thin feasible slice that the greedy baseline exhausts systematically but that ENAS's 30-sample Stage 1 covers only sparsely. Crucially, the search-time speedup is robust across the full range: every resolution gives a $\geq 1.29\times$ speedup, and the speedup grows with resolution on VWW (peaking at $3.11\times$ at 96×96) because NanoNAS's TFLite conversion cost scales with model size while ENAS's analytical pre-flight remains constant.

5.5. Pareto and Per-Hardware Decomposition

Fig. 3 places all 212 (method, dataset, hardware, input) cells on the accuracy–search-time plane; ENAS forms a tight low-cost cluster while NanoNAS spans a wider range that extends to higher accuracy at higher search time.

Table 6 and Fig. 4 decompose accuracy and search time across hardware. NanoNAS wins seven of eight per-hardware bests on VWW at 1.65–$3.48\times$ longer search time, with the largest advantages on mid-tier platforms (Pico, BLE, Nicla). ENAS wins decisively on STM32H743 ($+2.6$ pp at near-equal search time): the cell-based search space exposes a deep architecture with a k=13 stem reaching 79.4% TFLite accuracy at 80×80 (std 0.3 pp), the single best result in our study, and this advantage holds across multiple H743 resolutions ($+10.5$ pp at 32×32, $+10.6$ pp at 80×80). Because the search space admits stems up to $k = 16$, these wider high-capacity-MCU architectures lie entirely outside the NanoNAS (k, c) space.

Strategy vs. search space. To isolate the search strategy from the search space, we evaluated an intermediate variant (*ENAS-strategy*) retaining the original 2-D (k, c) space but using parallel random

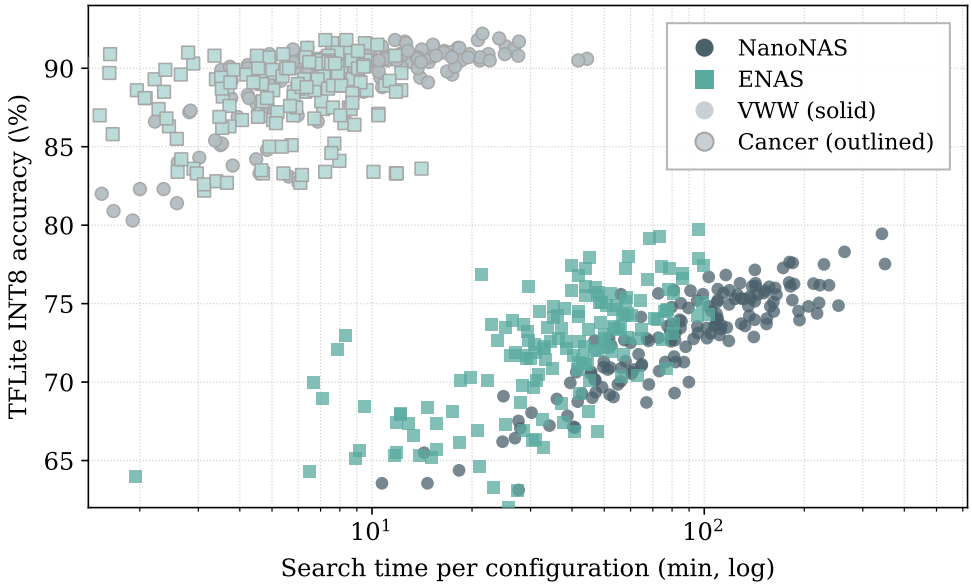

Figure 3: Accuracy versus search-time Pareto across all 212 cells. ENAS (teal squares) cluster at lower search times; NanoNAS (dark circles) spans a wider accuracy range. Cross-dataset stratification is visible: Cancer at top (∼89%), VWW at bottom (∼73%).

Table 6
Per-hardware best TFLite accuracy on VWW (mean±std over 3 runs at the best-accuracy resolution). "Px" is that resolution; "Speed" is the ENAS search-time advantage at the ENAS-best configuration.

HW	RAM	NanoNAS		ENAS		Speed
	(KB)	acc	Px	acc	Px	up
L010RBT6	20	72.7 ± 1.4	50	70.7 ± 3.4	32	2.44×
L010RB	20	72.0 ± 1.3	50	68.3 ± 3.1	32	2.79×
IoT	32	72.7 ± 1.4	32	70.4 ± 3.6	32	2.84×
L412	64	75.5 ± 0.6	80	74.0 ± 3.5	48	1.65×
Pico	264	75.9 ± 0.5	96	74.3 ± 1.9	50	3.48×
BLE	256	76.1 ± 1.1	128	75.6 ± 3.1	48	3.09×
Nicla	1024	77.1 ± 0.7	128	75.5 ± 1.5	50	3.02×
H743	1024	76.8 ± 1.4	96	79.4 ± 0.3	80	1.10×

search with the analytical pre-flight check. This variant captures the search-time speedup but none of the H743 accuracy gain, confirming that the cell-based search space drives the high-capacity-MCU advantage.

5.6. Ablation and Architectures

Table 7 reports the ablation. The dominant gain comes from the proxy-epoch change (+1.8/+2.2 pp), followed by accuracy-first scoring (+0.5 pp). Two-phase ranking and warm seed each contribute +0.3 pp; their value is greater on constrained devices where they prevent occasional pathological selections. Across the ENAS runs, depthwise-separable blocks dominate (48% of selected cells), followed by standard convolution (31%) and bottleneck (22%). The maximum Keras-to-TFLite INT8 drop across all ENAS runs is 0.2 pp, matching NanoNAS; `relu6` contributes by clipping activations into a quantisation-friendly range.

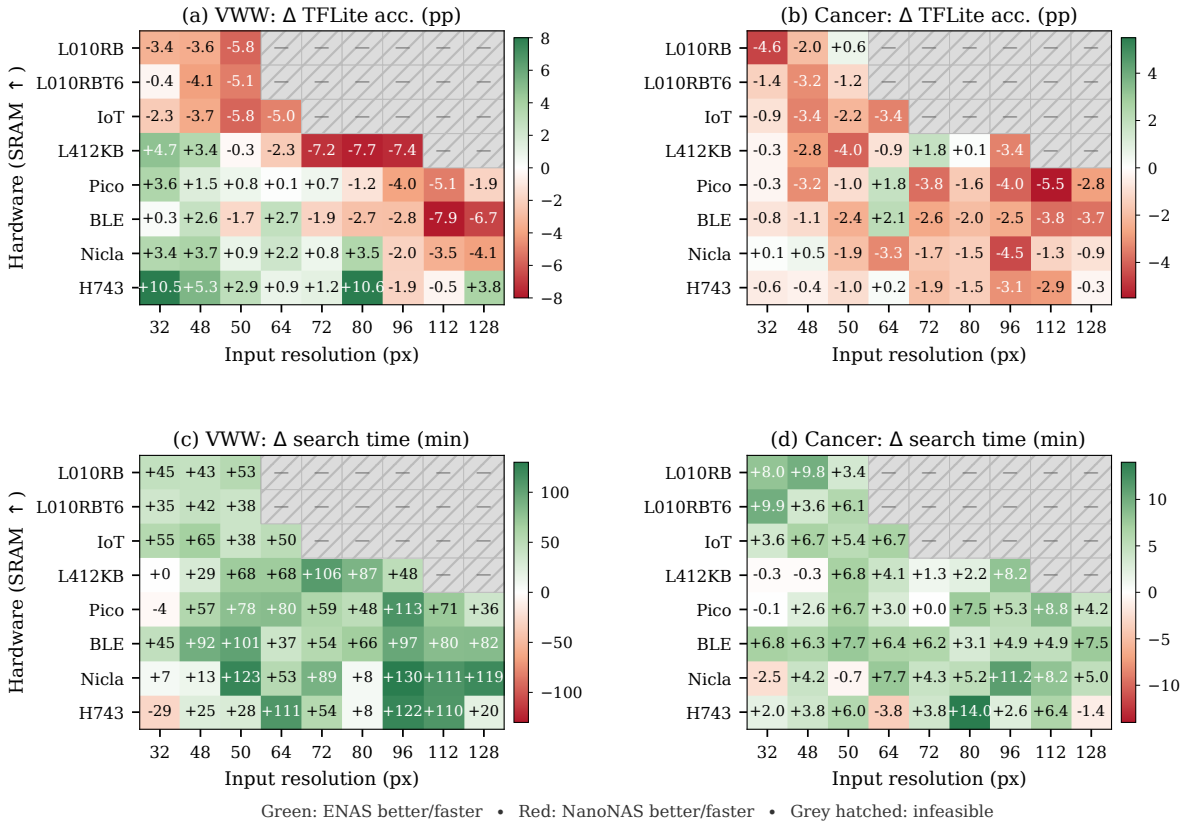

Figure 4: Per-cell deltas (ENAS vs. NanoNAS) across both datasets. Top row: Δ TFLite accuracy in percentage points (green = ENAS better). Bottom row: Δ search time in minutes (green = ENAS faster). The H743 row in panel (a) shows the strongest ENAS dominance (+10.5 pp at 32×32, +10.6 pp at 80×80); the search-time panels are almost uniformly green. Grey hatched cells are RAM-infeasible.

6. Discussion and Limitations

When to deploy ENAS. The accuracy–speed trade-off is hardware-tier-dependent. ENAS is the preferred method on high-capacity MCUs ($\geq$ 512 KB SRAM) at all input sizes, where the cell-based search space exposes architectures the 2-D (k, c) space cannot describe. On mid-tier hardware (64–256 KB) at large resolutions, NanoNAS retains a small mean-accuracy advantage, and the choice is dominated by the available search budget. On ultra-constrained hardware ($\leq$ 32 KB) at small resolutions, both methods perform similarly, with ENAS delivering the same accuracy at a 1.5–2× speedup. Independently of tier, ENAS is preferable when peak SRAM is scarce, since its selected models use ∼0.38× the activation RAM at matched accuracy (Section 5.2).

Sustainability and Search-Time Scaling. The entire empirical evaluation, encompassing two datasets, two optimization methods, eight microcontroller platforms, and nine input resolutions required approximately 297 CPU-hours on a single 16-core compute node, eliminating the need for GPU acceleration. To put this in perspective, MCUNet reports approximately 300 GPU-hours for a single VWW search instance [2]; our entire multi-dimensional sweep thus consumes less than a fraction of the carbon and computational footprint of a single MCUNet search point.

Furthermore, by bypassing the late-stage TFLite deployment bottleneck during optimization, the proposed framework successfully decouples input resolution from search-time scaling. As the resolution scales from 32×32 to 128×128, the average search time per configuration for ENAS increases by only 1.2× (1.20–1.25× empirically), demonstrating stable, sub-linear scaling characteristics. In contrast, NanoNAS exhibits a steeper 1.9× growth over the same range due to its continuous evaluation

Table 7
Ablation of design and calibration choices (mean VWW TFLite accuracy, %, at the two focused input sizes; single-configuration design progression). The proxy-epoch and scoring-weight choices together account for 2.4 pp; the two-phase ranking and warm seed each add a further 0.3 pp.

Variant	**Mean** 50×50	**Mean** 64×64
ENAS-strategy (2-D space)	70.2	71.5
ENAS, E_p=1, w_{acc}=0.65	68.8	70.5
+ E_p=3	70.6	72.7
+ w_{acc}=0.80	71.1	73.2
+ two-phase Stage 2	71.4	73.5
+ warm seed (full ENAS)	**71.7**	**73.7**

of larger activation tensors. This highly favorable scaling behavior positions ENAS as an accessible, environmentally sustainable choice for local edge laboratories lacking high-throughput cluster infrastructure.

Limitations and Future Work. Several open items remain for future investigation:
(i) Search budget limits: ENAS was not evaluated under an extended search budget matched to NanoNAS's longer runtime; whether additional candidates or proxy epochs close the large-resolution accuracy gap remains to be determined.
(ii) Modality constraints: The empirical validation is limited to image datasets; testing on audio, time-series, or sensor modalities [3, 17] is required to confirm cross-domain generality.
(iii) Baseline scope: This study excludes heavy GPU-based or supernet methodologies like MCUNet or ELASTIC [15] due to their differing cost regimes and task objectives.
(iv) Proxy validation: MACC counts serve as a latency and energy proxy; empirical on-board power measurements are needed to verify actual milliwatt-budget characteristics.
(v) Search space and mechanics: The cell-based search space is constrained relative to broader libraries like MBConv; additionally, the mutation mechanism discards infeasible candidates rather than resampling, and the search operates in a single pass without an outer convergence loop.
(vi) Flash estimation accuracy: The analytical Flash estimate under-predicts actual `stm32tflm` measurements. While sufficient here because RAM is the binding constraint, a calibrated estimator is necessary for deployments where Flash capacity binds.

7. Conclusion

This work demonstrates that lightweight, hardware-aware NAS for TinyML can successfully integrate an analytical pre-flight feasibility check with a cell-based search space and a three-stage hybrid optimization algorithm. Across two benchmarking datasets, eight MCU platforms, and nine input resolutions, this framework achieves consistent and statistically significant search-time speedups of 1.7–$2.4\times$ over NanoNAS. The associated accuracy trade-off is minor (-0.79 pp on VWW, which is not statistically significant; -1.31 pp on Cancer) and exhibits clear tier-dependent characteristics: ENAS outperforms alternative methods on high-capacity MCUs and within low-resolution regimes, whereas NanoNAS maintains an advantage on mid-tier hardware coupled with large input resolutions. Empirical resource analysis confirms that ENAS-selected architectures effectively trade Flash capacity and MACC operations for a substantial reduction in peak activation RAM the primary binding bottleneck in TinyML deployment at matched accuracy. These operational regimes are comprehensively characterized using 636 fully trained models, with all optimization logs open-sourced to ensure reproducibility. Ultimately, this framework provides a quantitatively mapped speed-accuracy-memory trade-off space, offering immediate utility for resource-constrained edge laboratories where minimizing both search overhead and deployment SRAM is paramount.

Acknowledgments

This work was supported by a research grant from the AI & Robotics Technology Park (ARTPARK) at the Indian Institute of Science.

Declaration on Generative AI

The authors used generative AI tools to assist with language editing, grammar, and formatting. All scientific content, experiments, and conclusions are the authors' own, and the authors take full responsibility for the content of this publication.

References

[1] A. M. Garavagno, E. Ragusa, A. Frisoli, P. Gastaldo, An affordable hardware-aware neural architecture search for deploying convolutional neural networks on ultra-low-power computing platforms, IEEE Sensors Letters 8 (2024) 1–4.

[2] J. Lin, W.-M. Chen, Y. Lin, C. Gan, S. Han, et al., Mcunet: Tiny deep learning on iot devices, Advances in neural information processing systems 33 (2020) 11711–11722.

[3] T. King, Y. Zhou, T. Röddiger, M. Beigl, Micronas for memory and latency constrained hardware aware neural architecture search in time series classification on microcontrollers, Scientific Reports 15 (2025) 7575.

[4] M. Tan, Q. Le, Efficientnet: Rethinking model scaling for convolutional neural networks, in: International conference on machine learning, PMLR, 2019, pp. 6105–6114.

[5] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 4510–4520.

[6] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, Q. V. Le, Mnasnet: Platform-aware neural architecture search for mobile, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 2820–2828.

[7] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, K. Keutzer, Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 10734–10742.

[8] H. Cai, L. Zhu, S. Han, Proxylessnas: Direct neural architecture search on target task and hardware, arXiv preprint arXiv:1812.00332 (2018).

[9] H. Cai, C. Gan, T. Wang, Z. Zhang, S. Han, Once-for-all: Train one network and specialize it for efficient deployment, arXiv preprint arXiv:1908.09791 (2019).

[10] M. S. Abdelfattah, A. Mehrotra, Ł. Dudziak, N. D. Lane, Zero-cost proxies for lightweight nas, arXiv preprint arXiv:2101.08134 (2021).

[11] J. Lin, W. Chen, H. Cai, C. Gan, S. Han, Mcunetv2: Memory-efficient patch-based inference for tiny deep learning (2021), Preprint arXiv 2110 (2021).

[12] J. Lin, L. Zhu, W.-M. Chen, W.-C. Wang, C. Gan, S. Han, On-device training under 256kb memory, Advances in Neural Information Processing Systems 35 (2022) 22941–22954.

[13] Y. Qiao, H. Xu, Y. Zhang, S. Huang, Micronas: Zero-shot neural architecture search for mcus, in: 2024 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, 2024, pp. 1–2.

[14] Y. Qiao, H. Xu, Y. Zhang, S. Huang, Monas: Efficient zero-shot neural architecture search for mcus, in: 2025 International Joint Conference on Neural Networks (IJCNN), IEEE, 2025, pp. 1–8.

[15] T. Tran, Q. Lin, B. Hu, Elastic: Efficient once for all iterative search for object detection on microcontrollers, IEEE Transactions on Computers (2026).

[16] A. M. Garavagno, D. Leonardis, A. Frisoli, Colabnas: Obtaining lightweight task-specific convolutional neural networks following occam's razor, Future Generation Computer Systems 152 (2024) 152–159.

[17] B. Saha, R. Samanta, S. K. Ghosh, R. B. Roy, Tinytnas: Gpu-free, time-bound, hardware-aware neural architecture search for tinyml time series classification, arXiv preprint arXiv:2408.16535 (2024).
[18] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, arXiv preprint arXiv:1704.04861 (2017).
[19] A. Chowdhery, P. Warden, J. Shlens, A. Howard, R. Rhodes, Visual wake words dataset, arXiv preprint arXiv:1906.05721 (2019).