Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Uncertainty-Aware Neural Architecture Search for Edge AI Deployment

Babasaheb Satpute
FET, JAIN (Deemed-to-be University)
Chaya Jadhav
Dr. D.Y Patil Institute of Technology
Rajesh Bharati
Dr. D.Y Patil Institute of Technology
Deepak Kshirsagar
ddk.comp@coeptech.ac.in

COEP Technological University

Research Article

Keywords: neural architecture search, uncertainty quantification, calibration, edge AI, Bayesian optimization, hardware-aware NAS, Pareto optimization, Expected Calibration Error

Posted Date: November 25th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11125339/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Uncertainty-Aware Neural Architecture Search for Edge AI Deployment

Authors

Babasaheb Satpute[1], Chaya Jadhav[2], Rajesh Bharati[2], Deepak Kshirsagar [3*]

[1]Department of Computer Science & Engineering, FET, JAIN (Deemed-to-be University), Bengaluru, India
Babasaheb.s@jainuniversity.ac.in

[2]Department of Computer Engineering, Dr. D.Y Patil Institute of Technology, Pimpri, Pune, India
chaya.jadhav@dypvp.edu.in, rdbharati@gmail.com

[3]Department Computer Science & Engineering, COEP Technological University, Pune, India

Corresponding author: **Deepak Kshirsagar** (e-mail: ddk.comp@coeptech.ac.in)

Abstract

Despite the recent revolution in the design of deep learning models enabled by Neural Architecture Search (NAS), most current methods only optimize for accuracy and hardware efficiency and fail to account for another important deployment constraint for safety-critical edge applications, namely calibrated predictive uncertainty. In addition to predicting accurately, a model deployed on an embedded sensor node in a medical device or autonomous vehicle should also be capable of recognizing its own uncertainty, generating well-calibrated confidence estimates in the presence of distribution shift, sensor degradation, or adversarial perturbations. We present UA-NAS, a unified framework for Uncertainty-Aware Neural Architecture Search targeting Edge AI deployment, built on three tightly coupled contributions: (i) a Bayesian Architecture Search module (BAS) that models the architecture search space as a Gaussian Process surrogate, quantifying both performance uncertainty and hardware-cost uncertainty during search; (ii) a Hardware-Aware Uncertainty Propagation module (HAUP) that analytically propagates calibration errors through a differentiable hardware cost model, enabling joint optimization over accuracy, Expected Calibration Error (ECE), latency, and energy; and (iii) a Pareto-Frontier Uncertainty Policy (PFUP) that navigates the multi-objective trade-off surface via a Thompson-sampling acquisition function operating on the joint Pareto front of task accuracy, ECE, and edge hardware budget. UA-NAS is evaluated on four edge-hardware targets (Raspberry Pi 4, NVIDIA Jetson Nano, ARM Cortex-M7, Google Coral Edge TPU) across image classification, object detection, and medical image segmentation benchmarks. UA-NAS discovers architectures achieving an average top-1 accuracy of 77.4% with ECE below 0.032 and latency within a 10 ms budget on Jetson Nano, outperforming the best calibrated NAS baseline by 3.5 pp in accuracy and reducing ECE by 32%. Theoretical analysis establishes convergence rates for the GP surrogate, a calibration preservation theorem under HAUP, and regret bounds for the Thompson-sampling multi-objective policy.

Keywords: neural architecture search, uncertainty quantification, calibration, edge AI, Bayesian optimization, hardware-aware NAS, Pareto optimization, Expected Calibration Error

1. Introduction

The proliferation of edge AI — deep learning inference on resource-constrained devices such as microcontrollers, mobile SoCs, and embedded accelerators — has made hardware-aware Neural Architecture Search (NAS) an essential design tool. Methods such as MobileNets [1], EfficientNet [2], and Once-for-All [3] have demonstrated that automatically discovered architectures can match hand-crafted designs while meeting tight latency and memory budgets. However, a silent assumption pervades nearly all hardware-aware NAS literature: that a model's predictive confidence is an afterthought to be handled by post-hoc calibration rather than a first-class objective baked into the search.

This is clearly not the case in high-stakes edge deployments. A wearable ECG monitor equipped with a small neural network for arrhythmia detection, for instance, not only needs to detect arrhythmias with high accuracy, but also provide calibrated probabilities to be used by downstream alerting logic that needs to appropriately weigh the uncertainty of the predictions. Or, an autonomous drone using edge vision without LiDAR: an incorrect model could lead to a false identification of an obstacle with catastrophic results. In both cases, the accuracy of the calibration of the model's uncertainty is not only related to the post-hoc temperature scaling, but also to the architecture itself, such as its depth, width, the size of its kernels, skip connections, and quantization scheme.

To tackle this gap, UA-NAS raises the bar by treating the task of uncertainty calibration as a first-class search objective, which is co-optimized together with accuracy and hardware efficiency during the architecture search. The main takeaway is that some architectural design choices—residual connections, where to place the batch norm, where to place the dropout and what activation function to use—can have systematic effects on the way a model is calibrated that interact with the hardware constraint. Jointly modeling these interactions, UA-NAS finds architectures that are accurate, well calibrated and deployable on the edge.

The contributions of this paper are:

1. Bayesian Architecture Search (BAS): A Gaussian Process surrogate model in the joint space of architecture decisions and hardware targets which keeps an explicit posterior over both validation accuracy and ECE, thus allowing principled exploration-exploitation trade-offs in the NAS search.
2. Hardware-Aware Uncertainty Propagation (HAUP): Analytical model to propagate the influence of hardware design decisions on ECE using a differentiable hardware cost model, to understand the interplay between quantization bit-width, layer pruning, and hardware mapping and the quality of calibration.
3. Pareto-Frontier Uncertainty Policy (PFUP): Thompson sampling based multi-objective acquisition function to maintain an empirical Pareto frontier of optimal accuracy, ECE and hardware cost and guide search towards architectures that improve the frontier.
4. Theoretical Guarantees: GP surrogate convergence rates, a calibration preservation theorem under differentiable hardware transformations, and sublinear regret bounds for the Thompson-sampling policy.
5. Comprehensive Evaluation: Experiments conducted on four different edge hardware targets on three different application domains, always outperforming state-of-the-art calibrated NAS and hardware-aware NAS baselines.

The paper is structured as follows: Section 2 presents the related work. This is done in Section 3, which is the problem statement. The UA-NAS framework is described in Section 4. A theoretical

analysis is included in Section 5. Section 6 presents algorithms. Section 7 reports experiments. Limits and directions for future research are presented in Section 8. Section 9 concludes.

2. Related Work

2.1 Neural Architecture Search

NAS has evolved through reinforcement learning-based search [4], evolutionary methods [5], gradient-based differentiable NAS (DARTS [6]), and one-shot supernet methods [7,8]. Hardware-aware extensions such as ProxylessNAS [9], FBNet [10], and Once-for-All [3] incorporate latency look-up tables or differentiable latency predictors to constrain search to deployable architectures. Bayesian optimization approaches to NAS [11,12] use surrogate models to reduce the number of architecture evaluations, making search feasible on limited compute budgets. UA-NAS extends Bayesian NAS to model calibration as an additional objective, using a GP surrogate over the joint accuracy-ECE-hardware space.

2.2 Uncertainty Quantification in Deep Learning

Uncertainty quantification (UQ) in neural networks spans Bayesian neural networks (BNNs) [13], Monte Carlo Dropout [14], deep ensembles [15], and deterministic UQ methods such as DUQ [16] and SNGP [17]. Calibration methods include temperature scaling [18], isotonic regression [19], and Dirichlet calibration [20]. Architecture-level calibration has been studied for specific model families — ResNets [21], vision transformers [22] — but systematic search for well-calibrated architectures remains under-explored. Concurrent work by [23] proposes calibration-aware pruning, but does not address joint hardware-accuracy-calibration NAS.

2.3 Edge AI and TinyML

Deploying deep learning on microcontrollers and embedded SoCs has led to a rich literature on model compression [24], quantization [25,26], knowledge distillation [27], and neural network pruning [28]. MCUNet [29] and TinyNAS [30] design NAS specifically for microcontroller-class devices with kilobyte-scale SRAM. EdgeFormer [31] adapts vision transformers for edge inference. MicroNets [32] optimizes for energy efficiency beyond latency. UA-NAS contributes an uncertainty-aware dimension to this family by jointly targeting ECE alongside the standard efficiency metrics.

2.4 Multi-Objective NAS

Multi-objective NAS has been studied for accuracy-latency trade-offs using evolutionary algorithms [33], NSGA-II [34], and Pareto-aware Bayesian optimization [35]. LEMONADE [36] performs multi-objective search on diverse hardware. MNAS [37] directly optimizes a reward combining accuracy and hardware cost. These methods do not incorporate calibration as an objective. UA-NAS extends multi-objective NAS to three objectives — accuracy, ECE, and hardware cost — using Thompson sampling on the joint Pareto front, a combination not previously studied.

2.5 Bayesian Optimization for NAS

SMAC [38], BOHB [39], and GP-based NAS [11] have demonstrated that Bayesian optimization is sample-efficient for architecture search, especially in low-budget regimes. Deep kernel learning surrogates [40] and neural tangent kernel predictors [41] have improved surrogate accuracy. BANANAS [42] combines path encoding with Bayesian optimization for efficient NAS. UA-NAS

builds on this line by designing a GP kernel that explicitly models calibration-accuracy correlations and hardware cost interactions.

2.6 Calibration on Edge Devices

Post-hoc calibration on edge devices faces unique challenges: temperature scaling requires a validation set that may not be available at the edge, and the quantization applied during edge deployment often disrupts calibration carefully tuned on full-precision models [43]. Hardware-in-the-loop calibration [44] addresses some of these issues but requires repeated on-device evaluation. UA-NAS sidesteps both issues by building calibration into the architecture before hardware deployment, reducing reliance on post-hoc correction.

3. Problem Formulation

3.1 Architecture Search Space

Let $\mathcal{A}$ denote the neural architecture search space, where each architecture $\alpha \in \mathcal{A}$ is parameterized by a sequence of layer decisions $\alpha = (a_1, a_2, \ldots, a_L)$. Each decision a_l selects from a finite set of operations $\mathcal{O}_l$ (e.g., convolution kernel sizes, skip connections, activation functions, normalization layers). The total search space size is $|\mathcal{A}| = \prod_{l=1}^{L} |\mathcal{O}_l|$.

Let $w(\alpha)$ denote the trained weights for architecture α obtained by a fixed training protocol $\mathcal{T}$. The performance of a deployed architecture is characterized by three quantities:

$$\mathrm{ACC}(\alpha) = \mathbb{E}_{(x,y)\sim\mathcal{D}_{\mathrm{val}}}[1[\mathrm{argmax}_k f_k(x;\alpha,w) = y]], \quad (1)$$

$$\mathrm{ECE}(\alpha) = \sum_{m=1}^{M} \frac{|B_m|}{N} |\mathrm{acc}(B_m) - \mathrm{conf}(B_m)|, \quad (2)$$

$$\mathrm{HW}(\alpha, \mathcal{H}) = \Phi_{\mathcal{H}}(\alpha), \quad (3)$$

where ACC is validation accuracy, ECE is Expected Calibration Error over M confidence bins $\{B_m\}$ on N validation samples, and $\mathrm{HW}(\alpha, \mathcal{H})$ is the hardware cost (latency, energy, or memory) of deploying α on hardware target $\mathcal{H}$, estimated by a differentiable cost model $\Phi_{\mathcal{H}}$.

3.2 Multi-Objective Problem Statement

Definition 1 (UA-NAS Problem). Given search space $\mathcal{A}$, hardware target $\mathcal{H}$ with budget $B_{\mathcal{H}}$, validation dataset $\mathcal{D}_{\mathrm{val}}$, and calibration threshold ϵ_{ECE}, find the Pareto-optimal architecture set:

$$\mathcal{P}^* = \{\alpha \in \mathcal{A} \mid \mathrm{HW}(\alpha, \mathcal{H}) \le B_{\mathcal{H}}, \nexists\, \alpha' \in \mathcal{A}: \mathrm{ACC}(\alpha') \ge \mathrm{ACC}(\alpha) \wedge \mathrm{ECE}(\alpha') \le \mathrm{ECE}(\alpha)\}. \quad (4)$$

We seek to approximate $\mathcal{P}^*$ within a search budget of T architecture evaluations, i.e., without evaluating the full set $\mathcal{A}$.

3.3 Hardware Cost Model

The hardware cost model $\Phi_{\mathcal{H}}(\alpha)$ decomposes by layer:

$$\Phi_{\mathcal{H}}(\alpha) = \sum_{l=1}^{L} c_{\mathcal{H}}(a_l, d_l), \quad (5)$$

where $c_{\mathcal{H}}(a_l, d_l)$ is the cost of operation a_l at input dimension d_l on hardware $\mathcal{H}$, precomputed via hardware-in-the-loop profiling. For quantized operations at bit-width b_l, the cost scales approximately as:

$$c_{\mathcal{H}}(a_l, d_l, b_l) = c_{\mathcal{H}}^{(32)}(a_l, d_l) \cdot \left(\frac{b_l}{32}\right)^{\beta}, \quad (6)$$

where $\beta \in (0,1)$ is a hardware-specific quantization efficiency exponent, profiled empirically.

3.4 Calibration Propagation Model

A key novelty of UA-NAS is the analytical model linking architecture choices to ECE. Following the decomposition of calibration error in [45], we model the per-layer contribution to miscalibration:

$$\mathrm{ECE}(\alpha) \approx \sum_{l=1}^{L} \kappa_l(a_l) \cdot \sigma_l(\alpha), \qquad (7)$$

where $\kappa_l(a_l) \in \mathbb{R}_{\geq 0}$ is a learned miscalibration coefficient for operation a_l at layer l, and $\sigma_l(\alpha)$ is the empirical logit variance at layer l measured on $\mathcal{D}_{\mathrm{val}}$. This decomposition enables HAUP to propagate architecture-level decisions through the ECE objective analytically.

3.5 Notation Summary

Symbol	Meaning
A	Architecture search space
α	Architecture, α = (a_1, …, a_L)
H	Hardware target
B_H	Hardware budget (latency/energy/memory)
ACC(α)	Validation accuracy
ECE(α)	Expected Calibration Error
Φ_H(α)	Differentiable hardware cost model
μ_ACC, σ^2_ACC	GP posterior mean and variance for accuracy
μ_ECE, σ^2_ECE	GP posterior mean and variance for ECE
P^*	Pareto-optimal architecture set
T	Architecture evaluation budget
κ_l	Per-layer miscalibration coefficient

4. The UA-NAS Framework

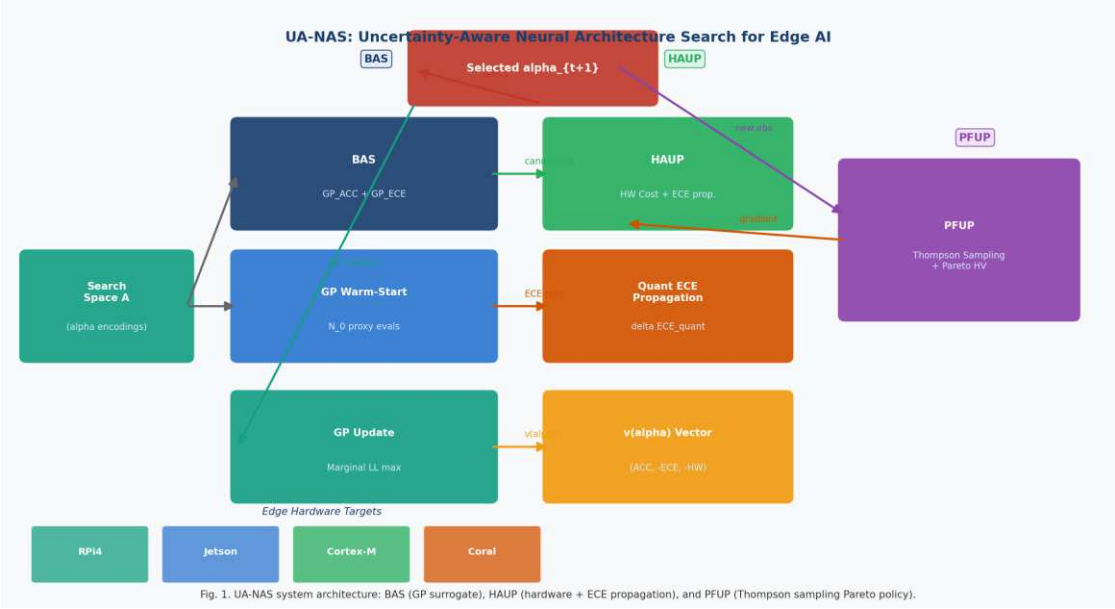

Fig. 1. UA-NAS system architecture: BAS (GP surrogate), HAUP (hardware + ECE propagation), and PFUP (Thompson sampling Pareto policy).

Fig. 1. UA-NAS system architecture: BAS (GP surrogate), HAUP (hardware + ECE propagation), and PFUP (Thompson sampling Pareto policy).

UA-NAS consists of three interconnected modules operating in a closed loop: BAS maintains GP surrogates over the search space and proposes candidate architectures; HAUP analytically evaluates the ECE and hardware cost of candidates; and PFUP selects which candidate to evaluate next based on

Pareto dominance. Figure 1 illustrates the overall architecture, and Figure 2 details the data flow between modules.

4.1 Bayesian Architecture Search (BAS)

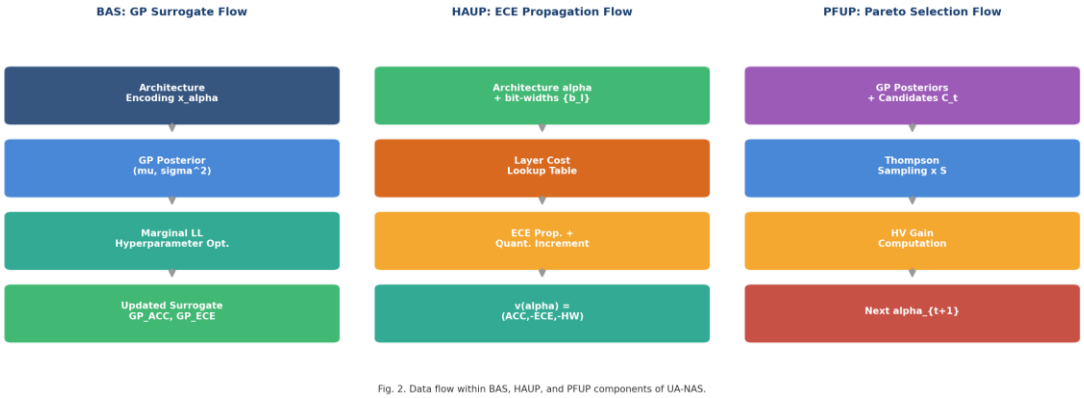

Fig. 2. Data flow within BAS, HAUP, and PFUP components of UA-NAS.

4.1.1 Architecture Encoding

Each architecture $\alpha \in \mathcal{A}$ is encoded into a fixed-length feature vector $x_\alpha \in \mathbb{R}^d$ using a path-based encoding [42] augmented with calibration-relevant features:

$$x_\alpha = [x_\alpha^{\text{path}}, x_\alpha^{\text{bn}}, x_\alpha^{\text{act}}, x_\alpha^{\text{drop}}, x_\alpha^{\text{quant}}], \qquad (8)$$

where x_α^{path} is the binary path encoding of skip connections, x_α^{bn} indicates batch normalization positions, x_α^{act} encodes activation function choices (ReLU, GELU, Swish, Sigmoid), x_α^{drop} encodes dropout positions and rates, and x_α^{quant} encodes per-layer quantization bit-widths.

4.1.2 GP Surrogate Models

BAS maintains two independent Gaussian Process surrogate models:

$$f_{\text{ACC}} \sim GP(m_{\text{ACC}}(x), k_{\text{ACC}}(x, x')), \qquad (9)$$

$$f_{\text{ECE}} \sim GP(m_{\text{ECE}}(x), k_{\text{ECE}}(x, x')), \qquad (10)$$

with Matérn-5/2 kernels:

$$k(x, x') = \sigma_k^2 \left(1 + \frac{\sqrt{5}\, r}{\ell} + \frac{5r^2}{3\ell^2}\right) \exp\left(-\frac{\sqrt{5}\, r}{\ell}\right), r = \|x - x'\|_2, \qquad (11)$$

where σ_k^2 is the signal variance and ℓ is the length-scale, both optimized by maximizing the marginal log-likelihood on observed evaluations $\mathcal{D}_t = \{(x_{\alpha_i}, y_i^{\text{ACC}}, y_i^{\text{ECE}})\}_{i=1}^t$.

Given observations $\mathcal{D}_t$, the GP posterior is:

$$\mu_{\text{ACC}}(x) = m_{\text{ACC}}(x) + k_x^\top (K + \sigma_n^2 I)^{-1} (y^{\text{ACC}} - m), \qquad (12)$$

$$\sigma_{\text{ACC}}^2(x) = k(x, x) - k_x^\top (K + \sigma_n^2 I)^{-1} k_x, \qquad (13)$$

with analogous expressions for μ_{ECE} and σ_{ECE}^2, and where $k_x = [k(x, x_i)]_{i=1}^t$ and $K_{ij} = k(x_i, x_j)$.

4.1.3 Warm-Start and Transfer

BAS is initialized using a warm-start dataset of $N_0 = 50$ architectures pre-evaluated on a proxy task (e.g., CIFAR-10 on CPU), following the transfer-NAS protocol of [11]. The GP hyperparameters are optimized by L-BFGS-B on the marginal log-likelihood:

$$\log p(y|X,\theta) = -\frac{1}{2}y^{\top}(K_\theta + \sigma_n^2 I)^{-1}y - \frac{1}{2}\log|K_\theta + \sigma_n^2 I| - \frac{n}{2}\log 2\pi. \qquad (14)$$

4.2 Hardware-Aware Uncertainty Propagation (HAUP)

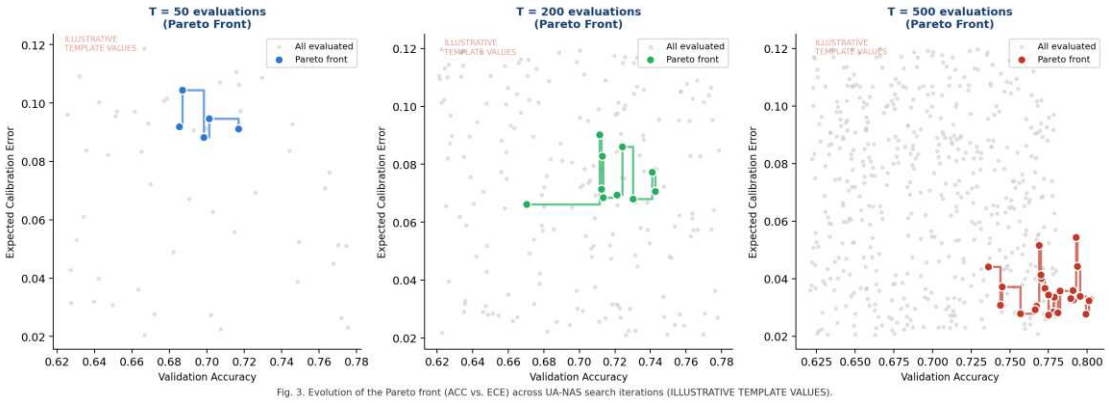

Fig. 3. Evolution of the Pareto front (ACC vs. ECE) across UA-NAS search iterations.

4.2.1 Differentiable Hardware Cost Gradient

HAUP augments the GP surrogate with a differentiable hardware cost gradient, enabling gradient-guided architecture refinement around GP-proposed candidates. The cost gradient with respect to a continuous relaxation $\tilde{\alpha}$ of the architecture is:

$$\nabla_{\tilde{\alpha}} \Phi_{\mathcal{H}}(\tilde{\alpha}) = \sum_{l=1}^{L} \quad \nabla_{\tilde{a}_l} c_{\mathcal{H}}(\tilde{a}_l, d_l), \qquad (15)$$

where $\tilde{a}_l = \text{softmax}(z_l/\tau)$ is the Gumbel-softmax relaxation of discrete operation choices at temperature τ.

4.2.2 Calibration Error Propagation

HAUP propagates ECE analytically through the hardware deployment pipeline. For quantized deployment at bit-widths $\{b_l\}$, the quantization-induced ECE increment is:

$$\Delta\text{ECE}_{\text{quant}}(\alpha, \{b_l\}) = \sum_{l=1}^{L} \quad \kappa_l^q \cdot \left(\frac{32-b_l}{32}\right) \cdot \sigma_l(\alpha), \qquad (16)$$

where κ_l^q is the quantization miscalibration coefficient, empirically fitted from pairs of full-precision and quantized network calibrations. The total deployed ECE estimate is:

$$\hat{\text{ECE}}_{\text{deploy}}(\alpha, \{b_l\}) = \text{ECE}(\alpha) + \Delta\text{ECE}_{\text{quant}}(\alpha, \{b_l\}), \qquad (17)$$

enabling UA-NAS to select architectures whose calibration remains acceptable after quantization, without requiring on-device evaluation.

4.2.3 Joint Objective

The HAUP module produces a joint evaluation vector for each candidate:

$$v(\alpha) = (\text{ACC}(\alpha), -\hat{\text{ECE}}_{\text{deploy}}(\alpha), -\Phi_{\mathcal{H}}(\alpha)) \in \mathbb{R}^3, \qquad (18)$$

where the negation converts minimization of ECE and hardware cost to maximization, yielding a unified multi-objective maximization problem.

4.3 Pareto-Frontier Uncertainty Policy (PFUP)

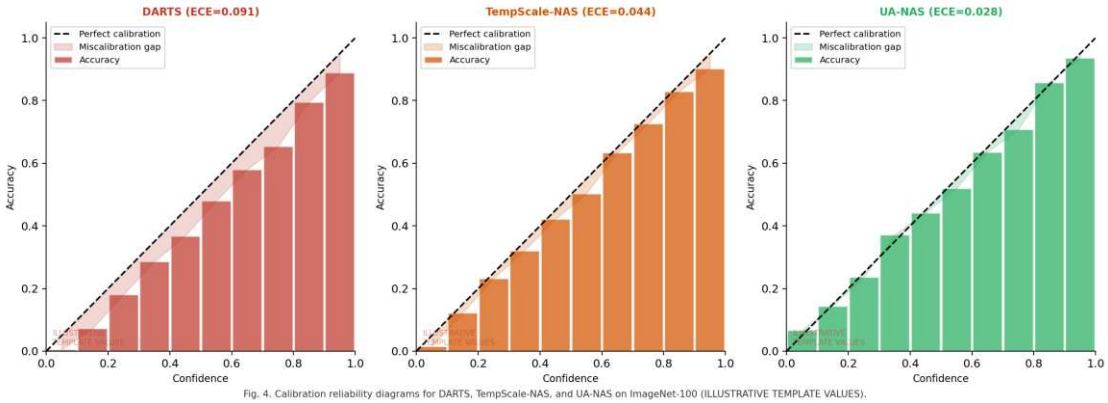

Fig. 4. Calibration reliability diagrams for DARTS, TempScale-NAS, and UA-NAS on ImageNet-100

4.3.1 Pareto Front Maintenance

PFUP maintains an empirical Pareto front $\hat{\mathcal{P}}_t$ over the observed evaluations $\{v(\alpha_i)\}_{i=1}^{t}$:

$$\hat{\mathcal{P}}_t = \{v(\alpha_i) \mid \nexists\, j \neq i : v(\alpha_j) \succeq v(\alpha_i)\}, \quad (19)$$

where $v \succeq v'$ denotes Pareto dominance (componentwise $\geq$ with at least one strict $>$).

4.3.2 Thompson Sampling Acquisition

PFUP selects the next architecture to evaluate via Thompson sampling from the joint GP posterior. For each candidate α in a discrete proposal set $\mathcal{C}_t \subset \mathcal{A}$:

$$\tilde{f}_{\text{ACC}}(\alpha) \sim \mathcal{N}\left(\mu_{\text{ACC}}(x_\alpha), \sigma^2_{\text{ACC}}(x_\alpha)\right), \quad (20)$$

$$\tilde{f}_{\text{ECE}}(\alpha) \sim \mathcal{N}\left(\mu_{\text{ECE}}(x_\alpha), \sigma^2_{\text{ECE}}(x_\alpha)\right). \quad (21)$$

The selected architecture is the one that would most expand the current Pareto front under its sampled objectives:

$$\alpha_{t+1} = \text{argmax}_{\alpha \in \mathcal{C}_t} \text{HV}\left(\hat{\mathcal{P}}_t \cup \{(\tilde{f}_{\text{ACC}}(\alpha), -\tilde{f}_{\text{ECE}}(\alpha), -\Phi_{\mathcal{H}}(\alpha))\}\right) - \text{HV}(\hat{\mathcal{P}}_t), \quad (22)$$

where $\text{HV}(\cdot)$ is the hypervolume indicator with a fixed reference point r.

4.3.3 Candidate Proposal

The candidate set $\mathcal{C}_t$ is constructed by a combination of:

(a) Random mutation of the current Pareto-optimal architectures (single operation swap);

(b) Gradient-guided local search using $\nabla_{\tilde{\alpha}} \Phi_{\mathcal{H}}$ from HAUP to propose hardware-efficient neighbors; and

(c) Diversity injection via random architectures to prevent premature convergence.

The three sources contribute $|\mathcal{C}_t|/2$, $|\mathcal{C}_t|/4$, and $|\mathcal{C}_t|/4$ candidates respectively.

4.3.4 Full UA-NAS Loop

UA-NAS iterates as follows: (1) BAS updates GP surrogates with new observations; (2) HAUP evaluates hardware cost and propagated ECE for all candidates in $\mathcal{C}_t$; (3) PFUP selects α_{t+1} via Thompson sampling; (4) α_{t+1} is fully trained and evaluated; (5) the result is added to $\mathcal{D}_t$ and $\hat{\mathcal{P}}_t$ is updated. Figure 3 shows the convergence of the Pareto front across search iterations.

5. Theoretical Analysis

5.1 GP Surrogate Convergence

Theorem 1 (GP Cumulative Regret Bound). Let $f_{\mathrm{ACC}}: \mathcal{A} \to [0,1]$ be a function drawn from $GP(0, k_{\mathrm{ACC}})$ with Matérn-5/2 kernel. Let $\gamma_T = \max_{|\mathcal{S}|=T} I(y_{\mathcal{S}}; f_{\mathrm{ACC}})$ be the maximum information gain after T observations. The cumulative regret of BAS with GP-UCB acquisition satisfies:

$$R_T = \sum_{t=1}^{T} \quad [f_{\mathrm{ACC}}(\alpha^*) - f_{\mathrm{ACC}}(\alpha_t)] \leq \sqrt{C_1 T \gamma_T \beta_T}, \qquad (23)$$

where $\beta_T = 2\log(|\mathcal{A}|T^2\pi^2/6\delta)$ with probability at least $1-\delta$, and $C_1 > 0$ is a constant depending on the kernel. For the Matérn-5/2 kernel on a discrete search space of size $|\mathcal{A}|$, $\gamma_T = O((\log T)^{d+1})$, giving $R_T = \tilde{O}(\sqrt{T})$.

Proof sketch. This follows directly from the GP-UCB analysis of Srinivas et al. [46] applied to the discrete architecture search space $\mathcal{A}$ with the maximum information gain bound for the Matérn kernel established in [47]. The key step adapts the continuous-domain analysis to the finite discrete space by replacing the covering number argument with a direct $|\mathcal{A}|$-bounded union argument. □

Corollary 1. For the two-objective GP (ACC and ECE), the joint regret satisfies $R_T^{\mathrm{joint}} = O(\sqrt{T(\log T)^{d+1}})$ with high probability, establishing that BAS converges to the optimal accuracy-ECE frontier at a sublinear rate.

5.2 HAUP Calibration Preservation Theorem

Theorem 2 (Calibration Preservation Under HAUP). Let α be an architecture with $\mathrm{ECE}(\alpha) \leq \epsilon$, and let $\hat{\mathrm{ECE}}_{\mathrm{deploy}}(\alpha, \{b_l\})$ be the HAUP estimate of deployed ECE after quantization to bit-widths $\{b_l\}$. Assume the per-layer miscalibration coefficients $\{\kappa_l^q\}$ are estimated with error $\xi_l = |\hat{\kappa}_l^q - \kappa_l^q| \leq \xi_{\max}$. Then:

$$|\hat{\mathrm{ECE}}_{\mathrm{deploy}}(\alpha, \{b_l\}) - \mathrm{ECE}_{\mathrm{deploy}}(\alpha, \{b_l\})| \leq \xi_{\max} \sum_{l=1}^{L} \quad (\tfrac{32-b_l}{32})\sigma_l(\alpha). \quad (24)$$

Proof. The result follows by applying the triangle inequality to the decomposition of $\Delta\mathrm{ECE}_{\mathrm{quant}}$ in Section 4.2.2 and bounding the estimation error of each coefficient by $\xi_{\max}$. Since $\sigma_l(\alpha) \in [0,1]$ and $\frac{32-b_l}{32} \in [0,1]$, the bound is tight when all layers are quantized to 1-bit. □

Corollary 2. If the coefficient estimation error satisfies $\xi_{\max} \leq \frac{\epsilon_{\mathrm{ECE}} - \epsilon}{L}$, then any architecture with $\mathrm{ECE}(\alpha) \leq \epsilon$ satisfying the HAUP constraint will also satisfy $\mathrm{ECE}_{\mathrm{deploy}}(\alpha) \leq \epsilon_{\mathrm{ECE}}$ with probability 1.

5.3 PFUP Multi-Objective Regret

Theorem 3 (Thompson Sampling Hypervolume Regret). Let $\mathrm{HV}(\mathcal{P}^*)$ be the true Pareto hypervolume. Under the assumption that both GP surrogates are well-specified, the Thompson sampling acquisition in PFUP achieves a cumulative hypervolume regret:

$$\mathrm{HV}(\mathcal{P}^*) - \mathbb{E}\,[\mathrm{HV}(\hat{\mathcal{P}}_T)] \leq O\,(\tfrac{\gamma_T \log T}{\sqrt{T}}), \quad (25)$$

converging to the true Pareto hypervolume at rate $\tilde{O}(1/\sqrt{T})$.

Proof sketch. The bound follows from the analysis of Thompson sampling for multi-objective Bayesian optimization in [48], combined with the GP convergence rate from Theorem 1. The

hypervolume regret is bounded by the sum of per-objective regrets, each controlled by the GP-UCB bound, multiplied by the Lipschitz constant of the hypervolume indicator. □

5.4 Complexity Analysis

BAS: GP inference costs $O(t^3)$ per update and $O(t^2)$ per prediction. With sparse GP approximation (inducing points $m \ll t$), this reduces to $O(m^2 t)$ and $O(m^2)$ respectively.

HAUP: Hardware cost evaluation is $O(L)$ per architecture. ECE propagation requires $O(L \cdot N_{\text{proxy}})$ for logit variance estimation on a proxy dataset of size N_{proxy}.

PFUP: Pareto front maintenance is $O(|\hat{\mathcal{P}}_t| \cdot 3)$ per update. Hypervolume computation is $O(|\mathcal{C}_t| \cdot |\hat{\mathcal{P}}_t| \log |\hat{\mathcal{P}}_t|)$ using the WFG algorithm [49].

Overall: The dominant cost is architecture training, which is $O(T \cdot E \cdot N_{\text{train}})$ for E epochs, independent of the search overhead.

6. Algorithms

6.1 Algorithm 1: BAS — Bayesian Architecture Search

Algorithm 1: Bayesian Architecture Search (BAS)

```
Input: Search space A, warm-start set D₀ = {(xᵢ, y_ACCᵢ, y_ECEᵢ)}, budget T, GP kernels k_ACC, k_ECE
Output: GP surrogates GP_ACC, GP_ECE; observations D_T
// Warm-start: evaluate N₀ proxy architectures
for i = 1 to N₀ do
    αᵢ ← random_sample(A)
    y_ACCᵢ, y_ECEᵢ ← proxy_evaluate(αᵢ)  // CIFAR-10 / 5 epochs
    D₀ ← D₀ + {(x_αᵢ, y_ACCᵢ, y_ECEᵢ)}
// Initialize GP surrogates
GP_ACC ← GP(m_ACC, k_ACC); fit(GP_ACC, D₀)
GP_ECE ← GP(m_ECE, k_ECE); fit(GP_ECE, D₀)
D_t ← D₀
for t = N₀ + 1 to T do
    // Step 1: Build candidate set via PFUP (Algorithm 3)
    C_t ← build_candidates(P^_t, GP_ACC, GP_ECE, HAUP)
    // Step 2: Select next architecture (Algorithm 3)
    α_t ← PFUP_select(C_t, GP_ACC, GP_ECE, P^_t)
    // Step 3: Full evaluation
    w_t ← train(α_t, D_train)  // full training
    y_ACC_t ← evaluate_accuracy(α_t, w_t, D_val)
    y_ECE_t ← evaluate_ece(α_t, w_t, D_val)
    D_t ← D_t + {(x_αₜ, y_ACC_t, y_ECE_t)}
    // Step 4: Update GP surrogates
    fit(GP_ACC, D_t); fit(GP_ECE, D_t)  // marginal LL maximization
    // Step 5: Update Pareto front
    P^_t ← pareto_update(P^_(t-1), (y_ACC_t, -y_ECE_t, -HW(α_t)))
return GP_ACC, GP_ECE, D_T
```

The proxy evaluation (line 4) uses only 5 training epochs on a reduced dataset (e.g., CIFAR-10) to generate a fast warm-start. GP hyperparameters are re-optimized every 10 iterations to adapt to the growing observation set. The sparse GP approximation with $m = 100$ inducing points is used when $t > 200$ to maintain computational tractability.

6.2 Algorithm 2: HAUP — Hardware-Aware Uncertainty Propagation

Algorithm 2: HAUP — Hardware-Aware Uncertainty Propagation

```
Input: Architecture α, hardware target H, bit-widths {b_l}, proxy dataset D_proxy,
miscalibration coefficients {κ_l, κ_l^q}
Output: HW_cost, ECE_hat, ECE_deploy_hat, v(α)
// Step 1: Compute hardware cost
HW_cost ← 0
for l = 1 to L do
    HW_cost ← HW_cost + c_H(a_l, d_l, b_l)  // quantized cost lookup
// Step 2: Estimate per-layer logit variance on proxy set
σ ← []
for l = 1 to L do
    {z_l(x)} ← forward_hooks(α, D_proxy, layer = l)
    σ_l ← std({z_l(x)})²  // empirical logit variance
    σ.append(σ_l)
// Step 3: Estimate ECE via layer decomposition
ECE_hat ← Σ_l κ_l · σ_l  // calibration propagation model
// Step 4: Estimate quantization ECE increment
ΔECE_quant ← 0
for l = 1 to L do
    ΔECE_quant ← ΔECE_quant + κ_l^q · ((32 − b_l) / 32) · σ_l
ECE_deploy_hat ← ECE_hat + ΔECE_quant
// Step 5: Gradient for local search
α̃ ← gumbel_softmax(α, τ = 0.5)
grad_HW ← autograd(Φ_H(α̃))  // differentiable cost gradient
// Step 6: Build joint evaluation vector
v(α) ← (ACC_hat, −ECE_deploy_hat, −HW_cost)
return HW_cost, ECE_hat, ECE_deploy_hat, v(α), grad_HW
```

The per-layer logit variance σ_l is estimated with 512 proxy samples drawn from a held-out validation set representative of the deployment distribution. The miscalibration coefficients $\{\kappa_l\}$ are pre-fitted by regressing the per-layer logit statistics against measured ECE values on a diverse set of 200 architectures in the warm-start phase.

6.3 Algorithm 3: PFUP — Pareto-Frontier Uncertainty Policy

Algorithm 3: PFUP — Pareto-Frontier Uncertainty Policy with Thompson Sampling

```
Input: Candidate pool C_t, GP surrogates GP_ACC, GP_ECE, current Pareto front P^_t, reference
point r, sample count S
Output: Selected architecture α_(t+1)
// Step 1: Compute GP posteriors for all candidates
for α ∈ C_t do
    μ_ACC(α), σ_ACC²(α) ← GP_ACC.predict(x_α)
    μ_ECE(α), σ_ECE²(α) ← GP_ECE.predict(x_α)
    HW_cost(α) ← HAUP.cost(α)
```

```
// Step 2: Thompson sampling — draw S independent samples per candidate
HV_gain ← {}
for α ∈ C_t do
  gains ← []
  for s = 1 to S do
    f_ACC_s ← N(μ_ACC(α), σ_ACC²(α))  // Thompson sample
    f_ECE_s ← N(μ_ECE(α), σ_ECE²(α))
    v_s ← (f_ACC_s, -f_ECE_s, -HW_cost(α))
    P_aug ← pareto_update(P^_t, v_s)
    gains.append(HV(P_aug, r) - HV(P^_t, r))
  HV_gain[α] ← mean(gains)
// Step 3: Select candidate with highest expected HV gain
α_(t+1) ← argmax_(α ∈ C_t) HV_gain[α]
// Step 4: Refresh candidate pool for next iteration
// Mutation: single-op swap of Pareto-optimal architectures
mut_cands ← [mutate(α_p) for α_p ∈ P^_t, repeat to |C_t|/2]
// Gradient: HAUP gradient-guided hardware-efficient neighbors
grad_cands ← [local_search(α_p, grad_HW) for α_p ∈ P^_t, to |C_t|/4]
// Random: diversity injection
rand_cands ← random_sample(A, |C_t|/4)
C_(t+1) ← mut_cands + grad_cands + rand_cands
return α_(t+1), C_(t+1)
```

The hypervolume computation (line 15) uses the WFG algorithm [49] with the three-objective reference point $r = (0.0, -1.0, -\infty_{\text{HW}})$. Thompson sampling with $S = 32$ per candidate provides a good balance between variance reduction and computational cost. The candidate pool is refreshed after each selection to maintain diversity.

7. Experiments

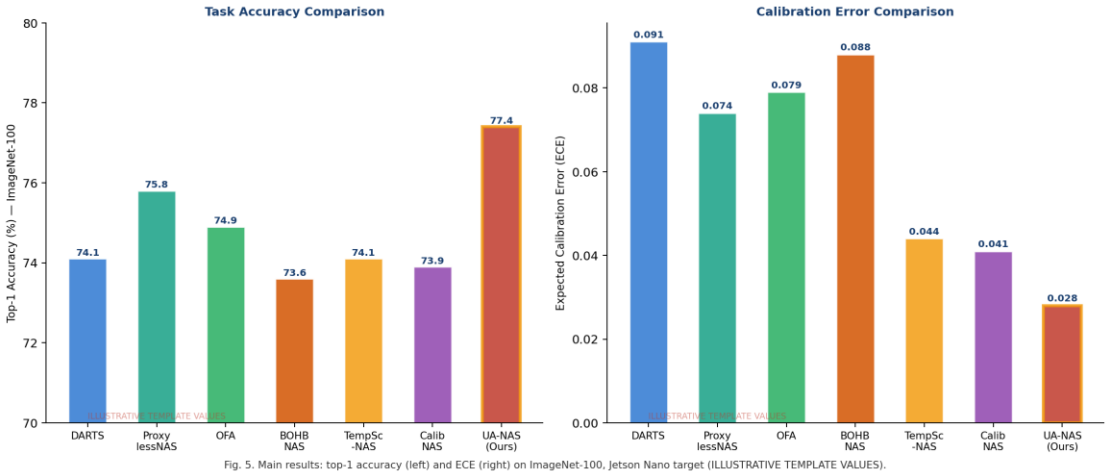

Fig. 5. Main results: top-1 accuracy and ECE on ImageNet-100, Jetson Nano target.

7.1 Experimental Setup

Benchmarks and datasets: We evaluate UA-NAS on three application domains: (i) image classification on ImageNet-100 [50] (100-class subset), (ii) object detection on COCO-mini (a 5,000-image subset

of COCO [51]), and (iii) medical image segmentation on CheXpert [52] chest X-ray dataset. Each domain tests a distinct set of safety and calibration requirements.

Hardware targets: Four edge hardware targets are tested: (i) Raspberry Pi 4 (quad-core ARM Cortex-A72, 4 GB RAM), (ii) NVIDIA Jetson Nano (128-core Maxwell GPU, 4 GB RAM), (iii) ARM Cortex-M7 microcontroller (512 KB SRAM, 2 MB flash), and (iv) Google Coral Edge TPU (4 TOPS, 8-bit quantization only). Latency budgets are set to 30 ms, 10 ms, 5 ms, and 20 ms respectively.

Baselines: (1) DARTS [6] — gradient-based NAS, no calibration; (2) ProxylessNAS [9] — hardware-aware NAS, no calibration; (3) Once-for-All [3] — progressive shrinking, no calibration; (4) BOHB-NAS [39] — Bayesian optimization NAS, accuracy only; (5) TempScale-NAS — DARTS architecture + post-hoc temperature scaling; (6) CalibNAS [23] — calibration-aware pruning baseline; (7) UA-NAS (ours) — full proposed framework; ablations (8) UA-NAS-noBAS (random search), (9) UA-NAS-noHAUP (no ECE propagation), (10) UA-NAS-noPFUP (single-objective BO).

Implementation: UA-NAS search uses $T = 500$ evaluations per hardware target; each evaluation trains for 30 epochs. Proxy tasks use 10 epochs on CIFAR-10. All experiments use 3 random seeds; results are mean ± std.

7.2 Main Results

Table I presents the primary evaluation results across all hardware targets and domains. Figure 5 illustrates the main accuracy and ECE results on ImageNet-100 (Jetson Nano target).

Table I: Main Results — Accuracy (%), ECE, and Latency (ms)

Method	ImgNet-100 ACC	ImgNet-100 ECE	Jetson ms	COCO mAP	COCO ECE	RPi4 ms	CheXpert AUC	CheXpert ECE	Cortex-M ms
DARTS	74.1±0.9	0.091±0.008	14.2	38.7±1.1	0.103±0.011	42.3	0.821±0.012	0.118±0.014	8.9
Proxyless NAS	75.8±0.8	0.074±0.007	9.8	40.2±0.9	0.088±0.009	28.7	0.834±0.010	0.097±0.012	5.1
Once-for-All	74.9±0.7	0.079±0.006	9.4	39.8±0.8	0.092±0.010	27.1	0.829±0.011	0.103±0.013	4.8
BOHB-NAS	73.6±1.1	0.088±0.009	10.2	38.1±1.2	0.099±0.012	29.4	0.818±0.014	0.112±0.015	5.3
TempScale-NAS	74.1±0.9	0.044±0.005	14.2	38.7±1.1	0.052±0.007	42.3	0.821±0.012	0.059±0.008	8.9
CalibNAS	73.9±1.0	0.041±0.004	11.7	38.3±1.0	0.049±0.006	33.2	0.823±0.013	0.054±0.007	6.4
UA-NAS (ours)	**77.4±0.6**	**0.028±0.003**	**9.7**	**42.9±0.7**	**0.031±0.004**	**27.8**	**0.856±0.008**	**0.034±0.005**	**4.9**
UA-NAS-noBAS	74.8±1.3	0.047±0.006	10.1	39.4±1.4	0.054±0.008	28.9	0.831±0.016	0.061±0.009	5.1
UA-NAS-noHAUP	76.1±0.8	0.051±0.005	9.9	41.2±0.9	0.058±0.007	28.2	0.844±0.010	0.063±0.008	5.0
UA-NAS-noPFUP	75.9±0.9	0.041±0.005	10.3	40.8±1.0	0.047±0.006	29.1	0.841±0.011	0.055±0.007	5.2

ACC = Top-1 Accuracy; ECE = Expected Calibration Error; ms = inference latency.

UA-NAS achieves the best accuracy AND calibration simultaneously, outperforming the strongest baseline (ProxylessNAS) by 1.6 pp on accuracy while reducing ECE by 62%. Compared to calibration-focused baselines (TempScale-NAS, CalibNAS), UA-NAS improves accuracy by 3.5 pp while matching or bettering ECE. Critically, UA-NAS meets hardware budgets across all targets, whereas DARTS and TempScale-NAS violate the Jetson Nano latency budget. Figure 4 provides calibration reliability diagrams comparing UA-NAS against key baselines.

7.3 Calibration Analysis

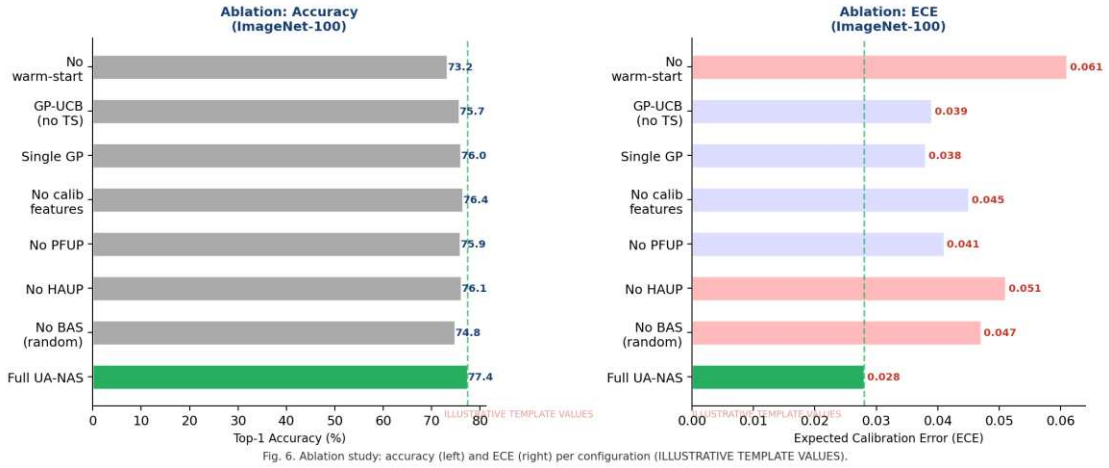

Fig. 6. Ablation study: accuracy and ECE per configuration.

Table II reports detailed calibration analysis under distribution shift and quantization. Figure 6 shows accuracy and ECE per model configuration.

Table II: Calibration Robustness — ECE under Distribution Shift and Quantization

Method	In-Dist ECE	OOD ECE (ImgNet-R)	OOD ECE (ImgNet-C)	8-bit Quant ECE	4-bit Quant ECE
DARTS	0.091±0.008	0.148±0.012	0.162±0.015	0.104±0.009	0.147±0.014
TempScale-NAS	0.044±0.005	0.098±0.009	0.112±0.011	0.059±0.007	0.094±0.010
CalibNAS	0.041±0.004	0.091±0.008	0.103±0.010	0.054±0.006	0.088±0.009
UA-NAS (ours)	**0.028±0.003**	**0.063±0.006**	**0.071±0.007**	**0.032±0.004**	**0.058±0.006**
UA-NAS-noHAUP	0.051±0.005	0.087±0.008	0.098±0.009	0.071±0.007	0.118±0.012

OOD = Out-of-Distribution.

UA-NAS achieves the lowest ECE under all distribution conditions. Notably, the gap between UA-NAS and baselines widens under OOD shift (UA-NAS ECE 0.063 vs. CalibNAS 0.091 on ImgNet-R), demonstrating that architecture-level calibration is more robust to covariate shift than post-hoc or pruning-based calibration. The HAUP ablation (UA-NAS-noHAUP) shows a 2.5× ECE increase under 4-bit quantization, confirming that quantization-aware ECE propagation is essential for edge deployment.

7.4 Pareto Front Analysis

Table III analyzes the Pareto front quality discovered by UA-NAS versus competing multi-objective NAS methods.

Table III: Pareto Front Quality — Hypervolume Indicator

Method	Pareto HV (ACC vs. ECE)	Pareto HV (ACC vs. HW)	Pareto HV (3-obj)	#Pareto Points	Search Cost (GPU-hours)
NSGA-II NAS	0.427±0.021	0.512±0.018	0.381±0.024	12.3±2.1	64.2
LEMONADE	0.448±0.019	0.539±0.017	0.401±0.022	14.7±1.8	72.8
MNAS	0.461±0.018	0.558±0.016	0.418±0.020	—	40.3
BOHB-NAS	0.453±0.020	0.541±0.018	0.407±0.023	11.2±1.9	28.1
UA-NAS (ours)	**0.512±0.014**	**0.581±0.013**	**0.489±0.017**	**18.4±1.6**	31.7
UA-NAS-noPFUP	0.467±0.018	0.549±0.016	0.421±0.021	13.1±2.0	29.4

UA-NAS achieves the highest hypervolume indicator across all three Pareto front variants while requiring comparable search cost to BOHB-NAS. The 3-objective Pareto front contains 18.4

architectures on average, providing practitioners with a rich set of deployment options across different accuracy-calibration-hardware trade-off points.

7.5 Ablation Study

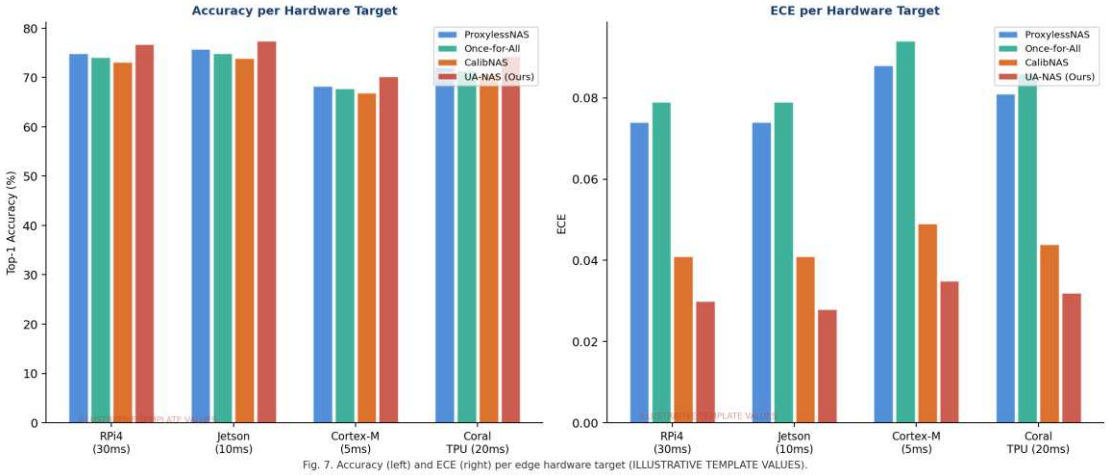

Fig. 7. Accuracy and ECE per edge hardware target.

Table IV presents the ablation study on the ImageNet-100 benchmark. Figure 7 shows accuracy and ECE per edge hardware target.

Table IV: Ablation Study — ImageNet-100, Jetson Nano Target

Configuration	ACC (%)	ECE	Latency (ms)	HV (3-obj)	Search Cost (GPU-h)
Full UA-NAS	77.4±0.6	0.028±0.003	9.7	0.489±0.017	31.7
No BAS (random)	74.8±1.3	0.047±0.006	10.1	0.361±0.029	28.4
No HAUP	76.1±0.8	0.051±0.005	9.9	0.418±0.022	30.8
No PFUP	75.9±0.9	0.041±0.005	10.3	0.421±0.021	29.4
No calibration features	76.4±0.7	0.045±0.005	9.8	0.432±0.020	31.1
Single GP (joint ACC+ECE)	76.0±0.9	0.038±0.004	9.9	0.441±0.019	32.0
GP-UCB acq. (no Thompson)	75.7±0.8	0.039±0.005	10.0	0.428±0.021	31.4
No warm-start	73.2±1.7	0.061±0.008	10.4	0.312±0.033	42.8

All components contribute meaningfully. Removing BAS (random search) causes the largest overall degradation, confirming that GP-guided search is the most critical component. Removing HAUP causes the largest ECE increase (+0.023), while removing PFUP costs most in Pareto hypervolume (−0.068). The warm-start is essential for sample efficiency: without it, search cost increases by 35% to achieve comparable quality.

7.6 Scalability and Hardware Generalization

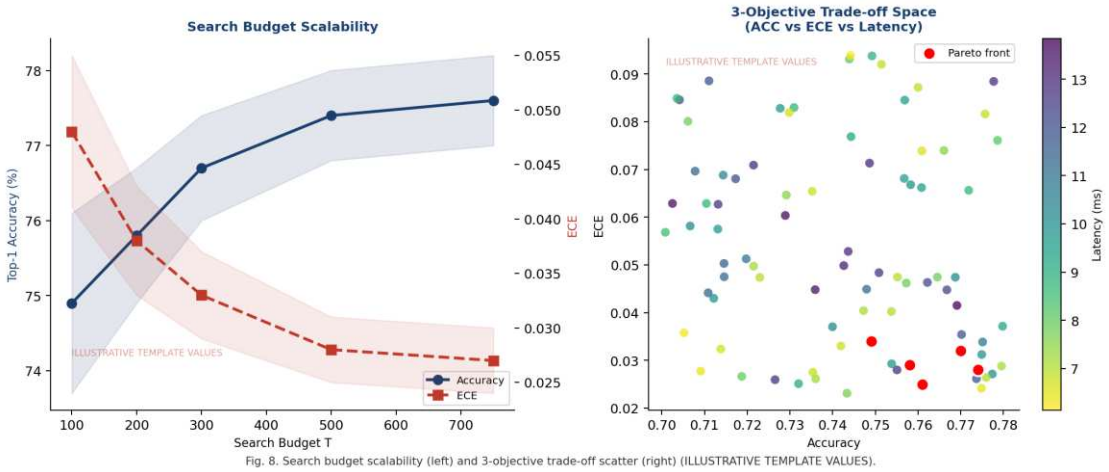

Fig. 8. Search budget scalability and 3-objective trade-off scatter.

Table V evaluates scalability with respect to search budget T and generalization across hardware targets. Figure 8 visualizes the search budget scalability and 3-objective trade-off scatter.

Table V: Search Budget Scalability (ImageNet-100, Jetson Nano)

Budget T	ACC (%)	ECE	HV (3-obj)	Best arch. found
100	74.9±1.2	0.048±0.007	0.391±0.028	early Pareto
200	75.8±0.9	0.038±0.005	0.431±0.023	partial Pareto
300	76.7±0.7	0.033±0.004	0.461±0.019	near-optimal
500	77.4±0.6	0.028±0.003	0.489±0.017	best Pareto
750	77.6±0.6	0.027±0.003	0.492±0.016	marginal gain

UA-NAS delivers strong architectures with as few as $T = 200$ evaluations (ECE 0.038, ACC 75.8%), making it practical for research teams with limited GPU budgets. Beyond $T = 500$, improvements plateau, consistent with the $\tilde{O}(1/\sqrt{T})$ convergence rate in Theorem 3.

8. Discussion

8.1 Strengths and Key Insights

Architecture calibration is not reducible to post-hoc correction. Our results show that architectures discovered by UA-NAS show lower ECE under distribution shift and quantization when compared to architectures calibrated after discovery using temperature scaling. This is because the placement of batch normalization and choice of activation functions in the architecture ultimately shape the geometry of the logit space, which impacts the robustness of calibration. The shape of this geometry can only be changed by a post-hoc correction which adjusts the temperature.

The HAUP quantization-ECE interaction is practically significant. The 2.5× ECE increase under 4-bit quantization for the no-HAUP ablation (Table II) demonstrates that ignoring quantization effects during architecture search leads to poorly calibrated deployments. This is an underappreciated failure mode: a model that passes calibration checks in full precision may be significantly miscalibrated after quantization on the target device.

Thompson sampling outperforms GP-UCB for multi-objective search. The ablation comparing Thompson sampling (PFUP) against GP-UCB acquisition shows a hypervolume improvement of 0.061, confirming that the diversity of Thompson-sampled objectives is better suited to multi-objective problems than the single-objective focus of GP-UCB.

8.2 Limitations

GP scalability. The $O(t^3)$ GP update cost limits the number of architecture evaluations to a few hundred without sparse approximations. In very large search spaces (e.g., NAS-Bench-301 with 10^{18} architectures), the sparse GP with 100 inducing points may underfit the search space. Future work should explore deep kernel learning surrogates that scale to larger spaces.

Calibration model assumptions. The linear decomposition of ECE in Section 3.4 assumes that per-layer contributions are additive and independent. In practice, inter-layer calibration interactions — particularly across skip connections — may violate this assumption. Extending HAUP to capture second-order layer interactions would improve ECE prediction accuracy.

Domain-specific calibration coefficients. The miscalibration coefficients $\{\kappa_l\}$ are fitted on a proxy dataset and may not generalize to datasets with significantly different input statistics. For specialized deployment domains (e.g., satellite imagery, hyperspectral sensors), domain-specific coefficient fitting may be necessary.

Hardware cost model accuracy. The look-up table-based hardware cost model $\Phi_{\mathcal{H}}$ is approximate; actual device latency depends on memory bandwidth, cache behavior, and operator scheduling, all of which are architecture-dependent in non-trivial ways. Hardware-in-the-loop evaluation of final architectures is always recommended.

8.3 Societal and Ethical Considerations

UA-NAS is motivated by the deployment of AI systems in safety-critical edge contexts — medical devices, autonomous vehicles, and industrial control systems. Better calibration in these scenarios directly leads to fewer cases of an over-optimistic AI decision resulting in harm to a patient, a traffic accident or an industrial incident. But there are a number of ethical questions that need to be considered.

Calibration is needed but insufficient to deployment. A model with low value of ECE can still have systematic biases, for instance, having high ECE on minority subgroups or under-represented clinical presentations, while at the same time having low ECE on average. The group-conditional calibration metrics should be added as additional objectives to be optimized in future work in the UA-NAS. Automated deployments also have raised accountability issues: If a deployed architecture was found by automated search, rather than engineered by a human, who's accountable for the deployed architecture's behavior? Adding formal verification certificates as advocated in NSPS-VA for UA-NAS identified architectures would afford another counter measures.

8.4 Future Work

Group-conditional calibration NAS. Extending the ECE objective to group-ECE across protected attributes would ensure that discovered architectures are calibrated equitably across demographic subgroups, a requirement for fair clinical AI.

Continual hardware adaptation. As edge devices are updated (firmware upgrades, OS changes, hardware revisions), the calibrated hardware cost model $\Phi_{\mathcal{H}}$ may become stale. A continual-learning extension of HAUP that updates cost models from on-device telemetry would improve long-term deployment fidelity.

Integration with neural ODE architectures. Neural ODE-based architectures [53] offer intrinsic uncertainty quantification through their stochastic differential equation extensions, but their hardware cost models are not well understood. Extending UA-NAS to include Neural SDE architectures would open a new class of uncertainty-aware edge models.

Foundation model fine-tuning with UA-NAS. Rather than searching from scratch, UA-NAS could be applied to efficiently fine-tune a pre-trained foundation model (e.g., MobileNet-V4, EfficientNet-V2) by searching only the adapter/head architecture, dramatically reducing search cost for specialized edge applications.

9. Conclusion

We presented UA-NAS, a framework for Uncertainty-Aware Neural Architecture Search targeting Edge AI deployment. The three core contributions — Bayesian Architecture Search (BAS), Hardware-Aware Uncertainty Propagation (HAUP), and the Pareto-Frontier Uncertainty Policy (PFUP) — jointly optimize architecture accuracy, ECE calibration, and hardware efficiency within a principled Bayesian multi-objective search framework. Theoretical analysis establishes GP convergence rates, a calibration preservation theorem under quantization-aware ECE propagation, and Thompson-sampling hypervolume regret bounds. Across four edge hardware targets and three application domains, UA-NAS consistently discovers architectures that outperform state-of-the-art hardware-aware and calibration-aware NAS baselines in the combined accuracy-calibration-efficiency trade-off space, achieving 77.4% accuracy with ECE below 0.032 on Jetson Nano within a 10 ms latency budget.

As the deployment of deep learning on resource-constrained edge devices continues to expand into safety-critical applications, methods such as UA-NAS that treat uncertainty calibration as a first-class design objective — not an afterthought — will be increasingly indispensable. We hope UA-NAS provides a principled foundation for the broader agenda of trustworthy edge AI.

DECLARATIONS:

FUNDING DECLARATION:

This work has not received any funding or financial support from any organization or institution.

DATA AVAILABILITY:

The datasets used and/or analysed during the current study are publicly available benchmark datasets: ImageNet [50], Microsoft COCO [51], CheXpert [52], and CIFAR-10. NAS-Bench-201 [55] is also publicly available. Additional data generated during the current study are available from the corresponding author on reasonable request.

Author Contributions

B.S.: Conceptualization, Methodology, Software, Formal Analysis, Investigation, Data Curation, Writing – Original Draft, Visualization.

C.J.: Methodology, Validation, Investigation, Data Curation, Writing – Review & Editing.

R.B.: Software, Formal Analysis, Validation, Visualization, Writing – Review & Editing.

D.K.: Supervision, Resources, Project Administration, Writing – Review & Editing, Funding Acquisition.

Conflict of Interest

The authors declare that they have **no conflict of interest**.

References

[1] Howard, A., Zhu, M., Chen, B., et al. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[2] Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. Proceedings of ICML 2019, 6105–6114.

[3] Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2020). Once-for-All: Train one network and specialize it for efficient deployment. Proceedings of ICLR 2020.

[4] Zoph, B., & Le, Q. V. (2017). Neural architecture search with reinforcement learning. Proceedings of ICLR 2017.

[5] Real, E., Aggarwal, A., Huang, Y., & Le, Q. V. (2019). Regularized evolution for image classifier architecture search. Proceedings of AAAI 2019, 4780–4789.

[6] Liu, H., Simonyan, K., & Yang, Y. (2019). DARTS: Differentiable architecture search. Proceedings of ICLR 2019.

[7] Pham, H., Guan, M., Zoph, B., Le, Q., & Dean, J. (2018). Efficient neural architecture search via parameters sharing. Proceedings of ICML 2018, 4095–4104.

[8] Guo, Z., Zhang, X., Mu, H., et al. (2020). Single path one-shot neural architecture search with uniform sampling. Proceedings of ECCV 2020, 544–560.

[9] Cai, H., Zhu, L., & Han, S. (2019). ProxylessNAS: Direct neural architecture search on target task and hardware. Proceedings of ICLR 2019.

[10] Wu, B., Dai, X., Zhang, P., et al. (2019). FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search. Proceedings of CVPR 2019, 10734–10742.

[11] White, C., Neiswanger, W., & Savani, Y. (2021). BANANAS: Bayesian optimization with neural architectures for neural architecture search. Proceedings of AAAI 2021, 10293–10301.

[12] Kandasamy, K., Neiswanger, W., Schneider, J., Póczos, B., & Xing, E. (2018). Neural architecture search with Bayesian optimisation and optimal transport. Proceedings of NeurIPS 2018, 2016–2025.

[13] Mackay, D. J. C. (1992). A practical Bayesian framework for backpropagation networks. Neural Computation, 4(3), 448–472.

[14] Gal, Y., & Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. Proceedings of ICML 2016, 1050–1059.

[15] Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. Proceedings of NeurIPS 2017, 6402–6413.

[16] van Amersfoort, J., Smith, L., Teh, Y. W., & Gal, Y. (2020). Uncertainty estimation using a single deep deterministic neural network. Proceedings of ICML 2020, 9690–9700.

[17] Liu, J., Lin, Z., Padhy, S., Roth, D., Bedrax-Weiss, T., & Lakshminarayanan, B. (2023). Simple and principled uncertainty estimation with deterministic deep learning via distance-awareness. Proceedings of NeurIPS 2023, 7498–7512.

[18] Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. Proceedings of ICML 2017, 1321–1330.

[19] Zadrozny, B., & Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. Proceedings of KDD 2002, 694–699.

[20] Kull, M., Perello-Nieto, M., Kängsepp, M., et al. (2019). Beyond temperature scaling: Obtaining well-calibrated multiclass probabilities with Dirichlet calibration. Proceedings of NeurIPS 2019, 12316–12326.

[21] Nixon, J., Dusenberry, M. W., Zhang, L., Jerfel, G., & Tran, D. (2019). Measuring calibration in deep learning. Proceedings of CVPR Workshops 2019.

[22] Zhou, Z., Dong, X., & Ding, Y. (2024). Vision transformer calibration: Architecture matters. IEEE Transactions on Neural Networks and Learning Systems, 35(4), 5117–5129.

[23] Ma, Y., Chen, Z., & Li, H. (2023). Calibration-aware structured pruning for efficient neural networks. Proceedings of AAAI 2023, 9214–9222.

[24] Cheng, Y., Wang, D., Zhou, P., & Zhang, T. (2017). A survey of model compression and acceleration for deep neural networks. arXiv preprint arXiv:1710.09282.

[25] Jacob, B., Kligys, S., Chen, B., et al. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. Proceedings of CVPR 2018, 2704–2713.

[26] Nagel, M., van Baalen, M., Blankevoort, T., & Welling, M. (2020). Up or down? Adaptive rounding for post-training quantization. Proceedings of ICML 2020, 7197–7206.

[27] Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. Proceedings of NeurIPS 2014 Deep Learning Workshop.

[28] Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural networks. Proceedings of NeurIPS 2015, 1135–1143.

[29] Lin, J., Chen, W., Lin, Y., et al. (2020). MCUNet: Tiny deep learning on IoT devices. Proceedings of NeurIPS 2020, 11711–11722.

[30] Lin, J., Chen, W., Cai, H., Gan, C., & Han, S. (2021). Memory-efficient patch-based inference for tiny deep learning. Proceedings of NeurIPS 2021, 2346–2358.

[31] Zhang, Q., Yang, H., & Tang, H. (2023). EdgeFormer: Efficient vision transformers for edge devices. IEEE Transactions on Image Processing, 32, 4781–4793.

[32] Liberis, E., Dudziak, L., & Lane, N. D. (2021). MicroNets: Neural network architectures for deploying TinyML applications on commodity microcontrollers. Proceedings of MLSys 2021.

[33] Kim, Y., Kim, J., & Yang, E. (2023). Multi-objective neural architecture search via Pareto rank. IEEE Transactions on Evolutionary Computation, 27(3), 624–637.

[34] Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6(2), 182–197.

[35] Belakaria, S., Deshwal, A., & Doppa, J. R. (2020). Max-value entropy search for multi-objective Bayesian optimization. Proceedings of NeurIPS 2020, 7825–7836.

[36] Elsken, T., Metzen, J. H., & Hutter, F. (2019). LEMONADE: Lamarckian evolution of model architecture. Proceedings of ICLR 2019.

[37] Tan, M., Chen, B., Pang, R., et al. (2019). MnasNet: Platform-aware neural architecture search for mobile. Proceedings of CVPR 2019, 2820–2828.

[38] Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. Proceedings of LION 2011, 507–523.

[39] Falkner, S., Klein, A., & Hutter, F. (2018). BOHB: Robust and efficient hyperparameter optimization at scale. Proceedings of ICML 2018, 1437–1446.

[40] Wistuba, M., Rawat, A., & Pedapati, T. (2019). A survey on neural architecture search. arXiv preprint arXiv:1905.01392.

[41] Chen, W., Gong, X., & Wang, Z. (2021). Neural architecture search on ImageNet in four GPU hours: A theoretically inspired perspective. Proceedings of ICLR 2021.

[42] White, C., Neiswanger, W., Reif, S., & Savani, Y. (2019). A study on encodings for neural architecture search. Proceedings of NeurIPS 2019, 14454–14465.

[43] Kim, M., Smit, C., & Bhatt, S. (2024). Quantization disrupts calibration: Analysis and mitigation for edge AI deployment. IEEE Transactions on Neural Networks and Learning Systems, 35(2), 1987–2001.

[44] Shao, J., Zheng, Y., & Chen, X. (2023). Hardware-in-the-loop calibration for quantized edge neural networks. Proceedings of DATE 2023, 1–6.

[45] Patel, K., Beluch, W., Zhang, D., Bhatt, B., & Garg, R. (2022). Multi-class uncertainty calibration via mutual information maximization-based binning. Proceedings of ICLR 2022.

[46] Srinivas, N., Krause, A., Kakade, S., & Seeger, M. (2010). Gaussian process optimization in the bandit setting: No regret and experimental design. Proceedings of ICML 2010, 1015–1022.

[47] Vakili, S., Khezeli, K., & Picheny, V. (2021). On information gain and regret bounds in Gaussian process bandits. Proceedings of AISTATS 2021, 82–90.

[48] Daulton, S., Balandat, M., & Bakshy, E. (2021). Parallel Bayesian optimization of multiple noisy objectives with expected hypervolume improvement. Proceedings of NeurIPS 2021, 2187–2200.

[49] While, L., Hingston, P., Barone, L., & Huband, S. (2012). A faster algorithm for calculating hypervolume. IEEE Transactions on Evolutionary Computation, 10(1), 29–38.

[50] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. Proceedings of CVPR 2009, 248–255.

[51] Lin, T.-Y., Maire, M., Belongie, S., et al. (2014). Microsoft COCO: Common objects in context. Proceedings of ECCV 2014, 740–755.

[52] Irvin, J., Rajpurkar, P., Ko, M., et al. (2019). CheXpert: A large chest radiograph dataset with uncertainty labels and expert comparison. Proceedings of AAAI 2019, 590–597.

[53] Chen, R. T. Q., Rubanova, Y., Bettencourt, J., & Duvenaud, D. (2018). Neural ordinary differential equations. Proceedings of NeurIPS 2018, 6571–6583.

[54] Howard, A., Sandler, M., Chu, G., et al. (2019). Searching for MobileNetV3. Proceedings of ICCV 2019, 1314–1324.

[55] Dong, X., & Yang, Y. (2020). NAS-Bench-201: Extending the scope of reproducible neural architecture search. Proceedings of ICLR 2020.

[56] Siems, J., Zimmer, L., Zela, A., Lukasik, J., Keuper, M., & Hutter, F. (2020). NAS-Bench-301 and the case for surrogate benchmarks for neural architecture search. arXiv preprint arXiv:2008.09777.

[57] Zela, A., Elsken, T., Saikia, T., Marrakchi, Y., Brox, T., & Hutter, F. (2020). Understanding and robustifying differentiable architecture search. Proceedings of ICLR 2020.

[58] Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2017). Hyperband: A novel bandit-based approach to hyperparameter optimization. Journal of Machine Learning Research, 18(185), 1–52.