Certified Multi-Device, Multi-Fidelity Neural Architecture Search for an ε-Pareto Front under Noisy and Drifting Hardware Measurements

Liz Lemma Future Detective

January 20, 2026

Abstract

Neural Architecture Search (NAS) has matured from reinforcement learning and evolutionary approaches to more efficient variants (e.g., differentiable and one-shot NAS) and hardware-aware optimization, but principled guarantees for multi-objective, deployment-faithful metrics remain scarce. Motivated by the growing 2026-era need to deploy foundation-model variants across heterogeneous platforms, we study NAS as a multi-objective identification problem: for each candidate architecture we seek a Pareto set trading off task quality with latency/token, energy/token, and memory footprint on multiple devices.

We formalize a multi-fidelity evaluation model that captures common NAS accelerators (early stopping, proxy tasks, weight-sharing) as biased low-fidelity oracles and treat hardware measurements as stochastic objectives with device-dependent noise and bounded drift. We propose MF-ParetoLUCB, an adaptive elimination algorithm that allocates expensive training and hardware profiling only to architectures whose Pareto status is uncertain. Our main results provide (i) a high-probability certificate that the returned set is an ε-Pareto approximation (inflated by an explicit drift term), (ii) a gap-dependent sample complexity upper bound on the number of high-fidelity evaluations, and (iii) a matching information-theoretic lower bound showing near-tightness. We also outline an instrumentation pipeline for repeated energy/token and latency/token measurement with drift detection, and propose experiments comparing against MONAS/DPP-style heuristics and predictor-only baselines to validate Pareto quality and stability in practice.

Table of Contents

2. 2. Related Work: efficient NAS (one-shot, differentiable), hardware-aware NAS and Pareto NAS (MONAS/DPP-Net), performance predictors, and multi-objective bandits/BO; positioning vs. NAS survey themes (computational cost, fairness/robustness, reproducibility).

3. 3. Problem Setup and Definitions: architecture set, objective vector across devices, Pareto optimality and ε-Pareto approximation; multi-fidelity biased-noisy oracle; hardware drift model; evaluation cost model and budget.

4. 4. MF-ParetoLUCB Algorithm: confidence bounds with bias + noise; Pareto-feasible set maintenance; ambiguous-point selection; fidelity promotion rules; stopping condition and output set; discussion of implementation choices.

5. 5. Main Theoretical Results: correctness (ε-Pareto with probability $1-\delta$); gap-dependent high-fidelity sample complexity; effect of drift as additive $+2\rho$; extensions to unknown β_k/ρ via calibration tests.

6. 6. Matching Lower Bounds: reduction from (multi-objective) best-arm identification; instance-dependent and minimax lower bounds; near-tightness discussion and when bounds can be improved with extra structure (e.g., Lipschitz descriptors).

7. 7. Instrumentation and Measurement Protocols (Experimental Spec): latency/token and energy/token measurement, repetitions, confidence intervals, change-point/drift detection; handling toolchain variability; how to compute per-token metrics and memory footprint consistently.

8. 8. Experimental Plan (Not Full Results): benchmarks (architecture search space, tasks, devices), baselines (MONAS/DPP, predictor-only, random + weight-sharing), evaluation metrics (hypervolume, Pareto regret, stability under reruns), ablations (no drift handling, no multi-fidelity, no elimination).

9. 9. Discussion and Limitations: dependence on finite $\mathcal{A}$, choice of β_k/ρ, practical proxy mismatch, extending to continuous search spaces or evolving operator sets; reproducibility checklist.

10. 10. Conclusion: certified Pareto NAS as a clean bridge from NAS heuristics to deployment-faithful guarantees.

1 Introduction

Neural architecture search (NAS) is now routinely performed under deployment constraints that differ materially from the regimes that motivated early work. In the 2026 deployment setting, a single model family is often expected to serve heterogeneous inference contexts: a datacenter GPU for batch throughput, an edge accelerator for on-device private inference, a mobile SoC for interactive generation, and occasionally a CPU-only fallback path. As a consequence, the relevant notion of "efficiency" is no longer captured by a single proxy such as parameter count or FLOPs, nor even by a single-device latency estimate. Instead, the operational cost of deploying an architecture is naturally described by per-token latency, per-token energy, and peak memory footprint on each target device, with each quantity measured (or estimated) in device-specific units and under device-specific runtimes.

This shift changes the NAS problem in three ways that we treat as structurally central.

First, the objective is intrinsically multi-objective and multi-device. Even if we fix a scalar notion of task performance (e.g., validation loss on a task suite), the deployment costs are vector-valued. For each device d, we must simultaneously control latency per token $\tau_d(a)$, energy per token $e_d(a)$, and memory $\mu_d(a)$. Aggregating these costs into a single scalar (via a weighted sum or a hand-designed penalty) is typically undesirable: it requires ex ante preference weights that differ across products and may change over time, and it obscures trade-offs that are relevant to downstream decision makers. We therefore treat the search target as a Pareto set under the full objective vector $f(a)$, which records quality together with the per-device costs.

Second, measuring these objectives is expensive and nonuniform in cost across coordinates and fidelities. High-fidelity quality evaluation for modern architectures may involve training or substantial fine-tuning, while high-fidelity device profiling requires running real kernels under representative runtimes and batch sizes, often with repeated measurements for noise reduction. In practice, one therefore uses a ladder of approximations: partial training, smaller data, smaller sequence lengths, simulators, compiler-level estimates, or surrogate device models. These approximations are cheaper but biased. Treating low-fidelity signals as merely noisy observations can lead to systematic errors in Pareto decisions. We hence adopt an explicit multi-fidelity measurement model in which an oracle query at fidelity k returns an observation $Y_k(a)$ whose bias is bounded in ℓ_∞ by a known radius β_k, while the remaining uncertainty is modeled as sub-Gaussian noise. The bias radii are assumed to be nonincreasing with k, with the top fidelity being unbiased.

Third, hardware measurements exhibit drift over time. Even when the underlying architecture is fixed, the effective observed objectives may shift

due to thermal conditions, background processes, runtime updates, compiler changes, or measurement protocol variations. Such drift is particularly salient for energy per token and latency per token, which can vary at the few-percent level in settings that are otherwise stable. Since Pareto comparisons are inherently relative, drift can induce inconsistent dominance relations if architectures are measured at different times. We therefore incorporate an explicit drift model: the effective objective vector at measurement time t may differ from a latent reference vector by at most ρ in ℓ_∞. This bound is intended to be either known (from an engineered measurement process) or estimated (from repeated measurement tests).

Under these conditions, the central algorithmic question is not merely how to discover good architectures, but how to *certify* that the returned set is an adequate approximation to the true Pareto set given finite budget, multi-fidelity bias, stochastic noise, and drift. We formalize the problem as follows. We are given a finite candidate set $\mathcal{A}$ and a finite set of devices $\mathcal{D}$. Each architecture $a \in \mathcal{A}$ induces a vector $f(a) \in \mathbb{R}^m$, where $m = 1 + 3|\mathcal{D}|$, and we aim to output a subset $\widehat{\mathcal{P}} \subseteq \mathcal{A}$ that is Pareto-accurate up to a specified additive tolerance ε. The algorithm may adaptively query a multi-fidelity oracle under a total budget constraint. The output is judged by a componentwise additive approximation: for every excluded architecture, there should exist a returned architecture that is no worse in every coordinate up to an additive slack. This additive criterion is the appropriate one in our setting because the coordinates are heterogeneous (loss, milliseconds/token, joules/token, and megabytes), and because multiplicative guarantees become ill-posed near zero after normalization.

Our approach is to treat the problem as a multi-objective pure-exploration task with structured uncertainty. The algorithm we analyze, MF-ParetoLUCB, maintains coordinatewise confidence bounds for each architecture, explicitly inflating them by the fidelity bias radius and the drift bound. Using these bounds, it maintains (i) a set of architectures that are still possibly nondominated, and (ii) certificates of elimination for architectures that are certainly ε-dominated (in a sense compatible with the inflated bounds). It then allocates further evaluation budget to those architectures whose Pareto status is ambiguous, choosing both which architecture to query and at what fidelity so as to decrease the relevant uncertainty at minimal cost. Low-fidelity evaluations are used whenever their bias radius is small relative to the remaining separation margin; otherwise, the algorithm promotes to higher fidelity. The algorithm stops when the remaining ambiguity is below the target tolerance or when the budget is exhausted.

We emphasize two aspects of the guarantee.

First, correctness depends on simultaneous validity of the maintained confidence regions over all architectures and all objectives. Since decisions are based on comparisons across many candidate vectors, pointwise concentration is insufficient: we require a uniform high-probability event under

which, for every architecture and objective coordinate, the true latent objective is contained in the maintained interval after accounting for bias and drift. This is achieved by standard sub-Gaussian concentration combined with an appropriate union bound over architectures, objectives, and update times, with deterministic inflation terms β_k and ρ added by triangle inequality. Conditioning on this event yields a purely deterministic argument: any elimination performed by the algorithm is safe, and any architecture excluded from the returned set is dominated up to the declared slack.

Second, drift imposes an unavoidable penalty. Even if the top fidelity is unbiased and the stochastic noise is small, two architectures measured at different times may experience worst-case drift in opposite directions. In the absence of additional synchronization assumptions, the best guarantee one can state in the worst case is an $(\varepsilon+2\rho)$-Pareto approximation. The factor 2ρ arises because the comparison of two architectures may involve two different measurement times, each deviating by up to ρ from the latent reference. Our analysis makes this dependence explicit and, conversely, clarifies how improved measurement protocols (e.g., time-synchronized profiling) would tighten the attainable approximation.

From the standpoint of sample efficiency, we are interested primarily in how many expensive, high-fidelity queries are required. In finite-set NAS, without additional structure, one cannot hope to avoid worst-case rates that scale linearly with $|\mathcal{A}|$, since any architecture could be Pareto-optimal. The more informative notion is instance-dependent complexity: architectures that are clearly dominated should be eliminated quickly, while those near the Pareto front require more careful measurement. We capture this via a Pareto gap Δ_a that quantifies the minimal ℓ_∞ slack by which a truly Pareto-optimal architecture can dominate a. Under a natural fidelity-promotion rule ensuring that bias is kept below the relevant separation margin, we obtain an upper bound on the expected number of highest-fidelity evaluations that scales as

$$\sum_{a \notin \mathcal{P}^*} \frac{\sigma_K^2}{(\Delta_a - \varepsilon)_+^2} \log \frac{nm}{\delta},$$

up to universal constants. This matches, up to constants and logarithmic factors, an information-theoretic lower bound obtained by a change-of-measure construction adapted to multi-objective dominance events. In this sense, the algorithm is near-optimal in its dependence on the instance-specific gaps.

We summarize our contributions.

Problem formalization for multi-device deployment. We formulate NAS as identification of an additive Pareto approximation under a full objective vector including quality and per-device latency, energy, and memory. This explicitly models the deployment regime in which a single architecture must be evaluated across a heterogeneous device set.

Multi-fidelity, costed evaluation model with bounded bias. We model practical NAS pipelines by permitting multiple fidelities with known costs, sub-Gaussian noise, and bounded bias radii. This enables principled use of cheap proxies without treating them as unbiased measurements.

Drift-aware certification. We incorporate bounded measurement drift and derive guarantees that degrade gracefully, showing precisely where and why the additive 2ρ penalty enters, and establishing that such a penalty is unavoidable without additional assumptions.

A certified adaptive algorithm and gap-dependent guarantees. We analyze MF-ParetoLUCB, which adaptively selects architectures and fidelities to resolve Pareto ambiguity. We provide high-probability correctness of an $(\varepsilon + 2\rho)$-Pareto approximation and derive gap-dependent bounds on the number of expensive top-fidelity evaluations, with a matching lower bound up to constants and logarithmic terms.

The remainder of the paper positions these results relative to prior work in NAS, hardware-aware optimization, and multi-objective pure exploration, and then develops the algorithm and theory in detail.

2 Related Work

We organize the related literature around four themes that bear directly on our setting: (i) computationally efficient NAS, (ii) hardware-aware and Pareto-oriented NAS, (iii) learned performance predictors and other proxy measurements (including multi-fidelity evaluation), and (iv) multi-objective pure exploration in bandits and Bayesian optimization. We close by noting connections to reproducibility and robustness themes emphasized in recent NAS surveys.

Efficient NAS: one-shot, weight sharing, and differentiable relaxations. A persistent bottleneck in NAS is the cost of evaluating candidate architectures. Early reinforcement-learning and evolutionary approaches reduced this cost only partially by amortizing search over many candidates, but still required training many models. This motivated *one-shot* and *weight-sharing* methods, in which a supernetwork (or over-parameterized model) is trained once and sub-architectures inherit weights for rapid evaluation. Representative families include ENAS-style approaches and subsequent variants that aim to reduce the bias introduced by shared weights, as well as once-for-all style methods that attempt to train a single network that supports many subnets. In parallel, *differentiable NAS* (e.g., DARTS and follow-ups) replaces discrete architecture choices with continuous relaxations so that architecture parameters can be optimized by gradient descent. These methods

greatly reduce wall-clock cost and have influenced the standard toolkit for practical NAS.

Our focus differs from this line in two respects. First, the central object in our analysis is not a continuous relaxation or a supernetwork, but a finite candidate set whose elements must be compared under a deployment-relevant objective vector. Second, while one-shot and differentiable methods target computational efficiency of search, they typically do not provide *certificates* for Pareto accuracy under measurement uncertainty, nor do they explicitly model the systematic bias of proxy objectives and the temporal drift of hardware measurements. In our setting, these issues are not second-order: low-fidelity evaluations (including weight-sharing estimates) can exhibit structured bias, and device profiling is susceptible to drift. We therefore treat proxy signals through explicit bias radii and maintain confidence regions that support safe elimination and correctness statements.

Hardware-aware NAS: constrained and cost-aware objectives. A second major line of work incorporates deployment constraints such as latency, memory, FLOPs, or energy into the NAS objective. A common approach is to solve a scalarized constrained optimization problem, for example by adding a penalty term for expected latency or enforcing a hard latency constraint during search. ProxylessNAS, FBNet-style methods, and MnasNet-type approaches are representative of this hardware-aware perspective, and have demonstrated that incorporating device-specific signals can substantially change the architecture choices favored by search. There is also a large body of work on compiler- and kernel-level modeling of latency, including lookup-table estimators, operator-level profiling, and analytic roofline-inspired approximations.

The limitation we address is that scalarized objectives require preferences or constraints fixed in advance, and they obscure trade-offs across multiple devices and cost types. In contrast, we treat the output as a set of architectures approximating the Pareto frontier under a vector of quality and per-device costs. This is closer to how deployment decisions are made in practice: one often wishes to preserve alternatives that trade off latency against energy differently on different devices, rather than committing to a single weighting. Moreover, the hardware-aware NAS literature frequently assumes that the measured cost (e.g., latency) is either deterministic or can be adequately handled as mean noise, whereas we explicitly model bounded bias (from simulators and surrogates) and bounded drift (from measurement conditions).

Pareto NAS and multi-objective evolutionary search. Multi-objective NAS has been studied extensively, often via evolutionary methods that maintain a population and use nondominated sorting, crowding-distance heuris-

tics, or explicit hypervolume contributions. MONAS and related approaches incorporate multiple objectives such as accuracy and latency; DPP-Net and similar methods produce Pareto sets and emphasize the diversity of returned architectures. Other works consider multi-objective variants of reinforcement learning or gradient-based approaches, frequently by scalarization or by optimizing a surrogate of hypervolume.

These approaches motivate our choice of a Pareto set as the appropriate output. However, the typical evaluation model in Pareto NAS implicitly assumes that objectives are either evaluated at a single fidelity or can be compared reliably after sufficient averaging. Our contribution is complementary: we isolate the *measurement* problem that arises once a candidate set is proposed (by any generator, including evolutionary or differentiable methods) and the practitioner must determine, under limited budget, which candidates belong to an approximate Pareto set across multiple devices. In particular, we treat multi-fidelity evaluation as first-class, allowing systematic proxy bias, and we incorporate drift into the certification logic.

Performance predictors, surrogates, and proxy measurements. A large body of work trains predictors for architecture performance, including accuracy predictors (trained from partial training runs, learning-curve extrapolation, or meta-models over architecture encodings) and hardware-cost predictors (latency/energy/memory regressors trained from profiling data). These predictors can be used within Bayesian optimization, evolutionary loops, or differentiable search pipelines to reduce the number of expensive evaluations. Multi-fidelity strategies are also common in practice: partial epochs, reduced datasets, smaller input resolutions, shorter sequence lengths, smaller batch sizes, quantized inference proxies, and simulators. In BO, multi-fidelity approaches such as MF-GP-UCB variants, FABOLAS-style models, and Hyperband/BOHB-type scheduling combine cheap approximate evaluations with selective promotion.

We share the high-level philosophy that cheap proxies should guide the allocation of expensive evaluations. The distinction is that we work in a setting where proxy errors are not well-modeled as mean-zero noise, especially for device-level measurements and short training runs. Instead, we represent proxy inaccuracy via an explicit ℓ_∞ bias radius that depends on the fidelity, and we integrate this radius into confidence bounds used for dominance decisions. This yields a different kind of guarantee than in typical surrogate-based NAS: rather than bounding regret or optimizing an expected scalar objective, we aim to return a set that is provably Pareto-accurate up to a declared additive tolerance (and drift). In this sense, our method can be viewed as providing a certification layer that can sit downstream of any candidate generation strategy and upstream of final deployment selection.

Multi-objective bandits, pure exploration, and Pareto front identification. On the theoretical side, our work is most closely connected to pure-exploration multi-armed bandits and best-arm identification, and to their extensions to multi-objective settings where the goal is to identify (approximately) the set of Pareto-optimal arms. Prior work in multi-objective bandits studies notions of Pareto regret, scalarization-based regret, and identification of nondominated sets under stochastic feedback. There is also a literature on LUCB-style algorithms and successive elimination for best-arm identification, as well as information-theoretic lower bounds based on change-of-measure and KL divergence arguments. In the multi-objective identification setting, analogous lower bounds arise because distinguishing dominance relations can require $\Theta(1/\Delta^2)$ samples when objective vectors are close.

Our analysis adapts this style of reasoning to the NAS measurement problem, with two additional complications that are less emphasized in standard multi-objective bandits: (i) multi-fidelity evaluations with systematic bias, and (ii) drift in the objective values over measurement time. Multi-fidelity pure exploration has been studied in bandits as well, often through hierarchical fidelity models or correlated observations, and with algorithms that decide when to promote to expensive unbiased feedback. We adopt a conservative bias-bounded model that yields simple, worst-case-valid certificates. Similarly, drift and nonstationarity are widely studied in online learning, but they typically enter through regret bounds or through explicit models of time variation; here drift appears as an irreducible penalty in a certification guarantee when comparisons are made across times.

Multi-objective Bayesian optimization and hypervolume search. Multi-objective BO is a natural alternative when $\mathcal{A}$ is large or continuous. It optimizes acquisition functions based on Pareto improvement, expected hypervolume improvement, or scalarizations sampled from a preference distribution. Multi-fidelity BO variants further incorporate evaluation cost and fidelity-dependent noise. These methods are powerful when smoothness or kernel structure can be exploited, and they provide a complementary perspective to our finite-set analysis. In our deployment setting, however, the candidate set is often discrete and already restricted by architectural constraints (e.g., kernel availability, compiler support, and product-specific design rules), and the dominant uncertainty is not only stochastic noise but also fidelity bias and drift. We therefore develop guarantees tailored to finite-set identification with conservative uncertainty inflation, rather than relying on smooth surrogate assumptions.

Reproducibility, robustness, and fairness considerations in NAS. Recent surveys and empirical studies in NAS emphasize challenges in reproducibility (e.g., sensitivity to implementation details, training protocols, and

search budgets), fairness of comparisons (e.g., differing evaluation budgets across methods), and robustness of conclusions under changes in data, random seeds, or hardware. Our drift model directly addresses one reproducibility failure mode for hardware-aware NAS: profiling results can shift over time due to uncontrolled factors, leading to inconsistent rankings and unstable Pareto sets. Likewise, our explicit bias-bounded multi-fidelity model addresses a fairness issue: methods that heavily exploit proxies may implicitly benefit from unaccounted systematic errors, whereas our framework requires that proxy usage be justified by conservative bounds. While our results do not solve all reproducibility concerns in NAS, they provide a formal mechanism for stating what can be certified from a given measurement protocol and budget, and for quantifying the degradation induced by drift without appealing to informal assumptions.

In summary, the existing NAS literature provides effective mechanisms for generating candidate architectures and for incorporating hardware costs, including multi-objective variants that output Pareto sets. The bandit and BO literatures provide principled tools for adaptive measurement under uncertainty. We synthesize these perspectives in a setting where the primary obstacle is *certified* Pareto identification under costed, biased multi-fidelity evaluations and time drift, and we analyze an adaptive algorithm whose guarantees match the instance-dependent lower bounds up to logarithmic factors.

3 Problem Setup and Definitions

We consider a finite set of candidate architectures $\mathcal{A}$ with $|\mathcal{A}| = n$, and a finite set of deployment devices $\mathcal{D}$ with $|\mathcal{D}| = D$. Each architecture $a \in \mathcal{A}$ is associated with a vector of deployment-relevant objectives that we seek to *minimize*. We separate a single quality objective from per-device cost objectives. Concretely, we write

$$f(a) \;=\; \big(\ell(a),\, (\tau_d(a),\, e_d(a),\, \mu_d(a))_{d\in\mathcal{D}}\big) \in \mathbb{R}^m, \qquad m \;=\; 1 + 3D,$$

where $\ell(a)$ denotes a task-quality loss (e.g., validation loss on a prescribed evaluation suite; equivalently, we may take $\ell(a) = -\text{accuracy}(a)$), and where for each device $d \in \mathcal{D}$ we include latency per token $\tau_d(a)$ (e.g., ms/token), energy per token $e_d(a)$ (e.g., J/token), and peak memory $\mu_d(a)$ (e.g., MB). We emphasize that device costs are inherently device-dependent and need not be correlated across d; our objective vector therefore explicitly retains all coordinates rather than collapsing them through scalarization.

Partial order and Pareto optimality. We compare objective vectors using the componentwise partial order $\preceq$ on $\mathbb{R}^m$: for $x, y \in \mathbb{R}^m$ we write

$x \preceq y$ if and only if $x_i \leq y_i$ for all coordinates $i \in [m]$. Architecture a is said to (weakly) dominate a' if $f(a) \preceq f(a')$. The (exact) Pareto set is then

$$\mathcal{P}^* \;=\; \big\{a \in \mathcal{A} :\; \nexists a' \in \mathcal{A} \text{ such that } f(a') \preceq f(a) \text{ and } f(a') \neq f(a)\big\}.$$

Since $\mathcal{A}$ is finite, $\mathcal{P}^*$ is nonempty. In applications we rarely expect to recover $\mathcal{P}^*$ exactly from finite and noisy measurements; accordingly, we adopt a standard additive approximation notion.

Additive ε-Pareto approximation. Fix $\varepsilon > 0$ and let $\mathbf{1} \in \mathbb{R}^m$ denote the all-ones vector. We say that $p \in \mathcal{A}$ *ε-dominates* $a \in \mathcal{A}$ if

$$f(p) \preceq f(a) + \varepsilon \mathbf{1}.$$

A subset $\widehat{\mathcal{P}} \subseteq \mathcal{A}$ is an ε-Pareto approximation if for every $a \notin \widehat{\mathcal{P}}$ there exists $p \in \widehat{\mathcal{P}}$ such that p ε-dominates a. This definition enforces coverage of all candidates by the reported set up to a uniform coordinatewise tolerance, which is natural when objectives have been normalized to commensurate scales (or when ε is specified per coordinate and absorbed into a weighted norm; here we use the uniform ℓ_∞ form for simplicity). Our goal is to identify such a set with high confidence while minimizing expensive evaluations.

Multi-fidelity evaluation oracle. We assume access to a costed oracle that can evaluate any architecture $a \in \mathcal{A}$ at any fidelity $k \in \{1, \ldots, K\}$. A query (a, k) returns an m-dimensional observation

$$Y_k(a) \;=\; f(a) + b_k(a) + \xi,$$

where $b_k(a) \in \mathbb{R}^m$ is a (possibly adversarial) fidelity-dependent bias vector and $\xi \in \mathbb{R}^m$ is a noise vector. We assume a known uniform bound on the bias magnitude,

$$\|b_k(a)\|_\infty \;\leq\; \beta_k \qquad \text{for all } a \in \mathcal{A},$$

where $\beta_k \geq 0$ is nonincreasing in fidelity k and $\beta_K = 0$. Thus fidelity K is unbiased, while lower fidelities may be systematically shifted. This model is intended to cover, for example, partial training runs (biased estimates of $\ell(a)$), learned predictors and simulators (biased estimates of $\tau_d(a)$, $e_d(a)$, $\mu_d(a)$), reduced sequence lengths, quantized proxy kernels, or other approximations that are not well-captured by mean-zero noise.

For the stochastic component, we assume that each coordinate of ξ is conditionally mean-zero sub-Gaussian with known proxy variance σ_k^2 at fidelity k (coordinatewise). Formally, conditioning on the history up to the query, for each coordinate $j \in [m]$ the scalar noise ξ_j satisfies the usual sub-Gaussian moment generating function bound with parameter σ_k^2. We do not require independence across coordinates; our analysis will rely only on coordinatewise concentration combined with a union bound across (a, j).

We allow the algorithm to adaptively choose the sequence of queries (a_t, k_t) based on all past observations. For a fixed architecture a, evaluations at different fidelities may be interleaved. In particular, we may initially screen many candidates using cheap biased fidelities and only later promote a smaller subset to high fidelity to resolve Pareto ambiguity.

Time drift and effective objectives. Hardware measurements (and, more generally, any end-to-end evaluation pipeline) may exhibit temporal drift due to thermal state, background processes, driver updates, frequency scaling, or measurement tool variability. To capture this, we introduce a time index t for measurement events and allow the effective objective vector at time t to be $f_t(a)$ rather than $f(a)$. We assume a bounded-drift condition: there exists $\rho \geq 0$ such that

$$\|f_t(a) - f(a)\|_\infty \;\leq\; \rho \qquad \text{for all } a \in \mathcal{A} \text{ and all measurement times } t.$$

Equivalently, $f(a)$ may be viewed as a nominal "time-averaged" objective vector and $f_t(a)$ as an adversarial but bounded perturbation. When drift is present, a query at time t should be interpreted as returning

$$Y_{k,t}(a) \;=\; f_t(a) + b_k(a) + \xi,$$

though in our notation we suppress the explicit dependence on t and incorporate ρ as an additional deterministic uncertainty radius. The drift model is deliberately conservative: it does not assume stationarity, smoothness over time, or shared drift structure across architectures. The only property used in certification is that two architectures evaluated at two different times can differ by up to 2ρ in their relative comparison, which naturally appears as an additive penalty in Pareto-approximation guarantees.

Evaluation cost model and budget constraint. Each query at fidelity k incurs a known nonnegative cost c_k. This cost is intended to aggregate all resources relevant for evaluation, including training time (for quality estimates), device profiling time (for latency/energy/memory), and any overhead from repetitions used to reduce measurement noise. We assume a total budget $B > 0$ and require that the algorithm selects a (random) sequence of queries $\{(a_t, k_t)\}_{t=1}^{T}$ satisfying

$$\sum_{t=1}^{T} c_{k_t} \;\leq\; B.$$

We make no structural assumption on $\{c_k\}$ beyond being known; in typical regimes, costs increase with fidelity, but this monotonicity is not needed for correctness (only for the efficiency interpretation of fidelity promotion). The central algorithmic question is then: how should we allocate a fixed evaluation budget across architectures and fidelities so as to return a certified approximate Pareto set?

Target guarantee and confidence parameter. Fix a failure probability $\delta \in (0,1)$. Our objective is to output a set $\widehat{\mathcal{P}} \subseteq \mathcal{A}$ such that, with probability at least $1-\delta$ over the algorithm's internal randomness and observation noise, every excluded architecture $a \notin \widehat{\mathcal{P}}$ is $(\varepsilon + 2\rho)$-dominated by some reported architecture $p \in \widehat{\mathcal{P}}$, i.e.,

$$\forall a \notin \widehat{\mathcal{P}} \;\; \exists p \in \widehat{\mathcal{P}} : \quad f(p) \preceq f(a) + (\varepsilon + 2\rho)\mathbf{1}.$$

The appearance of 2ρ is intrinsic to our drift model: even if two candidates are each estimated within ρ of their nominal vectors at their respective measurement times, their *difference* can be perturbed by up to 2ρ. Accordingly, we design our confidence regions by inflating statistical uncertainty with both the fidelity bias radii β_k and the drift radius ρ.

Gaps and instance difficulty (preview). For complexity statements it is convenient to quantify how "far" a suboptimal architecture lies from the Pareto frontier. One such notion is the (additive) Pareto gap Δ_a for $a \notin \mathcal{P}^*$, defined as the minimum ℓ_∞ slack required for some Pareto-optimal $p \in \mathcal{P}^*$ to dominate a on the coordinates where p is strictly better. While the precise form of Δ_a will matter only in our sample-complexity bounds, the informal message is that architectures very close to the frontier (small gap) are information-theoretically harder to classify, and therefore should be expected to consume the bulk of high-fidelity measurements.

With these definitions in place, we may now describe an adaptive procedure that maintains bias- and drift-inflated confidence bounds for each $f(a)$, uses these bounds to preserve a set of "possibly Pareto" candidates, and allocates evaluations across fidelities to eliminate candidates only when dominance is certified up to tolerance. This is the role of the MF-ParetoLUCB algorithm developed in the next section.

4 MF-ParetoLUCB: Multi-Fidelity Pareto Identification with Certified Elimination

We now describe MF-ParetoLUCB, an adaptive procedure that maintains (i) high-probability confidence boxes for each objective vector $f(a)$ under both fidelity bias and drift, and (ii) a set of architectures that are *not yet certifiably* ε-dominated. The algorithm allocates measurements to shrink those confidence boxes that presently obstruct a dominance certificate, promoting fidelity only when the residual bias at lower fidelities would prevent resolution.

Bias- and drift-inflated confidence bounds. For a fixed architecture a and coordinate $j \in [m]$, we may have collected samples at multiple fidelities.

Let $N_{a,k}(t)$ be the number of queries of (a,k) up to time t, and let $\overline{Y}_{a,k,j}(t)$ denote the corresponding empirical mean of the j-th coordinate. Using a standard sub-Gaussian tail bound with an anytime (or union-bounded) logarithmic factor, we choose radii $r_{a,k,j}(t)$ such that, simultaneously for all (a,k,j,t),

$$\left|\overline{Y}_{a,k,j}(t) - \mathbb{E}[\overline{Y}_{a,k,j}(t) \mid a,k]\right| \;\le\; r_{a,k,j}(t)$$

with probability at least $1-\delta$. One convenient concrete choice is

$$r_{a,k,j}(t) \;=\; \sigma_k \sqrt{\frac{2\log\!\left(\frac{4nmK(t+1)^2}{\delta}\right)}{\max\{1, N_{a,k}(t)\}}},$$

which suffices for a union bound over nmK streams and all $t \ge 1$. Since $\mathbb{E}[Y_{k,j}(a) \mid a,k] = f_j(a) + b_{k,j}(a)$ and $|b_{k,j}(a)| \le \beta_k$, and since drift contributes an additional ρ uncertainty in each coordinate, each fidelity k yields a valid coordinatewise interval

$$f_j(a) \in \Big[\overline{Y}_{a,k,j}(t) - r_{a,k,j}(t) - \beta_k - \rho,\; \overline{Y}_{a,k,j}(t) + r_{a,k,j}(t) + \beta_k + \rho\Big].$$

Because we may have multiple such intervals (across k), we combine them by intersection. That is, we define the global bounds

$$\begin{aligned}
\mathrm{LCB}_j(a,t) &:= \max_{k\in[K]:\, N_{a,k}(t)>0} \Big\{\overline{Y}_{a,k,j}(t) - r_{a,k,j}(t) - \beta_k - \rho\Big\},\\
\mathrm{UCB}_j(a,t) &:= \min_{k\in[K]:\, N_{a,k}(t)>0} \Big\{\overline{Y}_{a,k,j}(t) + r_{a,k,j}(t) + \beta_k + \rho\Big\},
\end{aligned}$$

with the convention that if a has not been measured then $\mathrm{LCB}(a,t) = -\infty\mathbf{1}$ and $\mathrm{UCB}(a,t) = +\infty\mathbf{1}$. The bounds for the full vector are $\mathrm{LCB}(a,t) = (\mathrm{LCB}_j(a,t))_{j\in[m]}$ and $\mathrm{UCB}(a,t) = (\mathrm{UCB}_j(a,t))_{j\in[m]}$. On the high-probability event from Theorem 5.1 (stated later), we have $f(a) \in [\mathrm{LCB}(a,t), \mathrm{UCB}(a,t)]$ componentwise for all a and all t.

Maintaining the set of possibly Pareto architectures. Given the confidence boxes, we use conservative dominance tests to (i) eliminate architectures that are already certifiably ε-dominated, and (ii) retain the remaining candidates as the current approximation set. At time t, define the set of *possibly nondominated* architectures by

$$\mathcal{P}_{\mathrm{cand}}(t) \;:=\; \Big\{a \in \mathcal{A} :\; \nexists a' \in \mathcal{A} \text{ such that } \mathrm{UCB}(a',t) \preceq \mathrm{LCB}(a,t) - \varepsilon\mathbf{1}\Big\}.$$

If $\mathrm{UCB}(a',t) \preceq \mathrm{LCB}(a,t) - \varepsilon\mathbf{1}$, then even under the most favorable realization for a and the most unfavorable realization for a' within their confidence boxes, a' still ε-dominates a. Thus such an a can be safely excluded from any $(\varepsilon + 2\rho)$-Pareto approximation on the confidence-valid event.

Dually, it is sometimes useful to explicitly track those candidates that are already *certainly dominated*:

$$\mathcal{D}_{\mathrm{cert}}(t) \;:=\; \Big\{ a \in \mathcal{A} :\; \exists a' \in \mathcal{A} \text{ such that } \mathrm{LCB}(a', t) \preceq \mathrm{UCB}(a, t) - \varepsilon \mathbf{1} \Big\}.$$

While $\mathcal{D}_{\mathrm{cert}}(t) \subseteq \mathcal{A} \setminus \mathcal{P}_{\mathrm{cand}}(t)$ need not hold as a strict identity for all t (due to the asymmetry of the tests), $\mathcal{D}_{\mathrm{cert}}(t)$ provides a convenient pool of architectures whose removal is already certified and hence need not be prioritized for further measurement.

Quantifying Pareto ambiguity. In order to decide what to measure next, we require a notion of which candidates are currently "blocking" elimination. We proceed as follows. For any ordered pair (a, a'), define a pairwise dominance slack functional

$$\Gamma_t(a', a) \;:=\; \max_{j \in [m]} \big(\mathrm{LCB}_j(a, t) - \mathrm{UCB}_j(a', t) \big).$$

If $\Gamma_t(a', a) > \varepsilon$, then $\mathrm{UCB}(a', t) \preceq \mathrm{LCB}(a, t) - \varepsilon \mathbf{1}$ and a is eliminated with witness a'. If $\Gamma_t(a', a) \leq \varepsilon$, the pair (a, a') remains ε-ambiguous in the sense that it is still *possible* (given the confidence boxes) that a' does not ε-dominate a. For a fixed a, we then define its worst-case "threat" level

$$\Gamma_t^\star(a) \;:=\; \min_{a' \in \mathcal{A}} \Gamma_t(a', a),$$

so that $\Gamma_t^\star(a) > \varepsilon$ implies that a can be eliminated, while $\Gamma_t^\star(a) \leq \varepsilon$ indicates that no elimination certificate is currently available. MF-ParetoLUCB prioritizes measurement of architectures in $\mathcal{P}_{\mathrm{cand}}(t)$ for which $\Gamma_t^\star(a)$ is small and whose confidence widths remain large, since these are precisely the architectures that might either (i) belong to the Pareto set, or (ii) be eliminated only after further refinement.

Selecting an architecture and objective coordinate to measure. Let the coordinatewise uncertainty width be

$$w_j(a, t) \;:=\; \mathrm{UCB}_j(a, t) - \mathrm{LCB}_j(a, t), \qquad w(a, t) \;:=\; \|w(a, t)\|_\infty = \max_{j \in [m]} w_j(a, t).$$

A simple and effective policy is to choose

$$a_t \in \arg \max_{a \in \mathcal{P}_{\mathrm{cand}}(t)} w(a, t),$$

i.e., we measure the candidate whose confidence box is widest in ℓ_∞. In addition, we may focus on the coordinate $j_t \in \arg\max_j w_j(a_t, t)$, interpreting the next measurement as one that should reduce the dominant source of uncertainty. In the present oracle model, a query returns the full vector $Y_k(a)$, so the coordinate selection is only used to guide the fidelity decision; in extensions with partial observations (e.g., separate profiling calls per device), one may query only the necessary coordinate group.

Fidelity choice and promotion. Having selected a_t, we choose a fidelity k_t by comparing the *attainable* uncertainty reduction at each fidelity against its cost. The key point is that for a fixed k, even with infinite repetitions the best possible half-width cannot drop below $\beta_k + \rho$ (and below ρ when $k = K$). Thus lower fidelities become uninformative once β_k exceeds the resolution required to decide ε-dominance.

A concrete promotion rule is the following. Let $\widehat{\Delta}_t(a_t)$ be an empirical "margin-to-decision" proxy, for instance

$$\widehat{\Delta}_t(a_t) \;:=\; \varepsilon - \Gamma_t^\star(a_t),$$

clipped to $[0, \varepsilon]$. If $\widehat{\Delta}_t(a_t)$ is small, then only modest refinement is needed; if it is large (meaning $\Gamma_t^\star(a_t)$ is far below ε), then a_t is deeply ambiguous and any useful refinement must be substantial. We then select the cheapest fidelity k such that

$$\beta_k \;\le\; \frac{1}{4}\max\{\widehat{\Delta}_t(a_t),\, \varepsilon\},$$

and sample at that fidelity, promoting to higher k only when this constraint fails for all cheaper fidelities. The constant $1/4$ is not essential; it encodes the principle that bias should consume only a controlled fraction of the remaining decision margin so that statistical sampling can close the gap. In particular, if a_t remains ambiguous even after extensive low-fidelity sampling, the rule forces promotion because $r_{a,k,j}(t)$ decreases with $N_{a,k}(t)$ while β_k does not.

Stopping condition and output. MF-ParetoLUCB may stop either upon budget exhaustion or upon certified resolution. A sufficient resolution condition is that every excluded architecture $a \notin \mathcal{P}_{\mathrm{cand}}(t)$ has an explicit witness $p \in \mathcal{P}_{\mathrm{cand}}(t)$ with

$$\mathrm{UCB}(p, t) \preceq \mathrm{LCB}(a, t) - \varepsilon\mathbf{1},$$

and, simultaneously, that the remaining ambiguity among candidates is ε-small in the sense that for all $a, a' \in \mathcal{P}_{\mathrm{cand}}(t)$ we do not have $\mathrm{UCB}(a', t) \preceq \mathrm{LCB}(a, t) - \varepsilon\mathbf{1}$. When this condition holds, no further eliminations are possible without violating the confidence-valid event, and the natural output is

$$\widehat{\mathcal{P}} \;:=\; \mathcal{P}_{\mathrm{cand}}(t).$$

If the budget is exhausted before the stopping rule triggers, we output the current $\mathcal{P}_{\mathrm{cand}}(t)$ as well; correctness is then interpreted as an $(\varepsilon + 2\rho)$-Pareto approximation on the confidence-valid event (the analysis in Section 5 makes this precise).

Implementation considerations. First, the intersection-based bounds $\mathrm{LCB}, \mathrm{UCB}$ naturally support heterogeneous data collection: one may mix

cheap, biased simulators for device costs with expensive on-device measurements, and similarly mix partial training runs with full training for quality, without assuming that lower fidelities are linearly related to higher ones. Second, although maintaining $\mathcal{P}_{\text{cand}}(t)$ by naive pairwise checks costs $O(n^2m)$ per update, in practice $m = 1 + 3D$ is moderate and the candidate set typically shrinks rapidly; moreover, incremental maintenance is possible by re-testing dominance relations only for architectures whose confidence boxes changed. Third, drift enters only through the additive inflation ρ. If the experimenter can periodically re-measure a small anchor set of architectures to estimate or upper bound ρ, then the same algorithm applies with the estimated bound (we return to calibration extensions after stating the main results). Finally, MF-ParetoLUCB is compatible with repeated measurements and averaging at a fixed (a, k): increasing $N_{a,k}(t)$ shrinks $r_{a,k,j}(t)$ at the usual $1/\sqrt{N}$ rate, and the promotion logic ensures that the algorithm does not waste repetitions at a fidelity whose bias floor β_k already exceeds the remaining Pareto decision tolerance.

In summary, MF-ParetoLUCB maintains a conservative set of possibly Pareto architectures using dominance certificates derived from bias- and drift-inflated confidence boxes, and it adaptively invests evaluation budget to shrink the few confidence boxes that presently obstruct such certificates, promoting fidelity only when necessary. The next section states the formal correctness guarantee and the resulting gap-dependent high-fidelity complexity bounds.

5 Main Theoretical Results

We now state the formal guarantees for MF-ParetoLUCB. Throughout, we interpret correctness as a *certified* additive Pareto approximation under the componentwise order $\preceq$, and we measure complexity primarily by the number of highest-fidelity evaluations ($k = K$), since these are typically the dominant cost drivers. Our analysis separates three effects that must be handled simultaneously: statistical noise (captured by σ_k), fidelity bias (captured by β_k), and measurement drift (captured by ρ).

5.1 Uniform confidence under bias and drift

The first step is to guarantee that, with probability at least $1 - \delta$, every true objective vector $f(a)$ lies in its maintained confidence box for all times t. Because the algorithm adaptively chooses which (a, k) to query, we require a uniform (anytime) concentration guarantee over the entire adaptive transcript. Since we only assume conditional sub-Gaussianity, a standard approach is to apply a tail bound to each empirical mean and then take a union bound over (a, k, j) and a suitably discretized (or summed) time index. The deterministic bias and drift then enter through triangle inequalities.

Theorem 5.1 (Uniform Confidence with Bias and Drift)**.** *Fix $\delta \in (0,1)$. Suppose that for each fidelity $k \in [K]$ and each coordinate $j \in [m]$, the noise ξ_j is conditionally mean-zero sub-Gaussian with proxy variance σ_k^2, and that $\|b_k(a)\|_\infty \le \beta_k$ for all $a \in \mathcal{A}$. Assume also that drift satisfies $\|f_t(a) - f(a)\|_\infty \le \rho$ for all a, t. Define radii $r_{a,k,j}(t)$ so that*

$$\Pr\Big(\forall a, k, j, t:\ |\overline{Y}_{a,k,j}(t) - \mathbb{E}[\overline{Y}_{a,k,j}(t) \mid a, k]| \le r_{a,k,j}(t)\Big) \ge 1 - \delta,$$

for instance using an anytime union bound. Then, on the same event, for all $a \in \mathcal{A}$, $j \in [m]$, and t,

$$f_j(a) \in \big[\mathrm{LCB}_j(a,t),\ \mathrm{UCB}_j(a,t)\big],$$

where LCB, UCB *are the intersection-based bounds defined from $\overline{Y}_{a,k,j}(t)$ by inflating with $\beta_k + \rho$.*

The proof is immediate once one observes that the oracle expectation equals $f_j(a) + b_{k,j}(a)$ while the effective measured value at time t may deviate from $f_j(a)$ by at most ρ. Consequently, each empirical mean produces a valid interval after adding $\beta_k + \rho$, and the intersection over fidelities preserves validity.

5.2 Certified $(\varepsilon + 2\rho)$-Pareto correctness

We next translate uniform confidence into a Pareto approximation guarantee for the returned set. The essential point is that dominance decisions are made by comparing two architectures using (potentially) measurements collected at different times; even if we were to query at unbiased fidelity $k = K$, drift can move the effective objective vectors in opposite directions at those two times. This yields an unavoidable additive slack 2ρ in the worst case. The algorithm is therefore analyzed with the target ε internal to the dominance tests, while the final approximation guarantee becomes $\varepsilon + 2\rho$ when stated with respect to the reference f.

Theorem 5.2 ($(\varepsilon + 2\rho)$-Pareto Correctness)**.** *Assume the confidence-valid event of Theorem 5.1 holds. Let $\widehat{\mathcal{P}}$ be the set output by MF-ParetoLUCB, i.e., the final $\mathcal{P}_{\mathrm{cand}}(t)$ at the stopping time (or budget exhaustion time). Then for every $a \notin \widehat{\mathcal{P}}$ there exists $p \in \widehat{\mathcal{P}}$ such that*

$$f(p) \preceq f(a) + (\varepsilon + 2\rho)\mathbf{1}.$$

Equivalently, $\widehat{\mathcal{P}}$ is an $(\varepsilon + 2\rho)$-Pareto approximation of $\mathcal{A}$ under f.

A proof sketch proceeds by the elimination rule: if $a \notin \mathcal{P}_{\mathrm{cand}}(t)$, then by definition there exists some witness $p \in \mathcal{A}$ such that $\mathrm{UCB}(p,t) \preceq \mathrm{LCB}(a,t) - \varepsilon\mathbf{1}$. On the confidence-valid event, we have $f(p) \preceq \mathrm{UCB}(p,t) + \rho\mathbf{1}$ and

$f(a) \succeq \mathrm{LCB}(a,t) - \rho\mathbf{1}$ componentwise, since the bounds were inflated by ρ to cover temporal mismatch. Combining these inequalities yields

$$f(p) \preceq \mathrm{UCB}(p,t) + \rho\mathbf{1} \preceq \mathrm{LCB}(a,t) - \varepsilon\mathbf{1} + \rho\mathbf{1} \preceq f(a) - \varepsilon\mathbf{1} + 2\rho\mathbf{1},$$

which is the claimed $(\varepsilon + 2\rho)$ dominance. If the algorithm stops by certified resolution, we may further ensure that $\widehat{\mathcal{P}}$ contains all architectures that are not $(\varepsilon + 2\rho)$-dominated; if it stops by budget exhaustion, the statement above still holds because elimination certificates are never invalidated on the confidence-valid event.

The preceding theorem makes explicit why drift impacts the final approximation additively. In fact, without further assumptions (e.g., synchronized measurement times or a parametric drift model), a 2ρ penalty is minimax-unavoidable: any comparison between two items measured at two times can suffer a worst-case discrepancy of ρ in each direction.

5.3 Gap-dependent high-fidelity sample complexity

We now quantify how many expensive highest-fidelity evaluations are required. The relevant instance-dependent quantity is the Pareto gap Δ_a for each $a \notin \mathcal{P}^*$, which captures how far a lies from being nondominated. When Δ_a is large, a can be eliminated with relatively coarse confidence boxes (and hence typically without high fidelity); when Δ_a is small, many samples may be needed, and promotion to unbiased fidelity is necessary once the low-fidelity bias floor would exceed the resolution required.

For notational compactness, define $(x)_+ := \max\{x, 0\}$. We state a representative bound for the number of $k = K$ queries under a promotion rule that ensures that, whenever a's status is unresolved, we only use a fidelity k whose bias radius β_k is at most a fixed fraction of the remaining decision margin. Concretely, it suffices that for such a the algorithm eventually promotes until $\beta_k \le (\Delta_a - \varepsilon)/4$, as this guarantees that bias cannot mask ε-separation.

Theorem 5.3 (Gap-Dependent High-Fidelity Complexity Upper Bound)**.** *Assume the confidence-valid event of Theorem 5.1. Suppose MF-ParetoLUCB promotes fidelity so that any $a \notin \mathcal{P}^*$ that remains in $\mathcal{P}_{\mathrm{cand}}(t)$ beyond a constant number of updates is sampled only at fidelities k satisfying $\beta_k \le (\Delta_a - \varepsilon)/4$, and ultimately at $k = K$ if needed. Then the expected number N_K of highest-fidelity queries satisfies*

$$\mathbb{E}[N_K] = O\left(\sum_{a \notin \mathcal{P}^*} \frac{\sigma_K^2}{(\Delta_a - \varepsilon)_+^2} \log \frac{nm}{\delta}\right),$$

up to universal constants. Moreover, the total expected cost admits an analogous bound that weights contributions by the costs c_k at the fidelities actually used.

The argument parallels LUCB-style elimination analyses: each suboptimal $a \notin \mathcal{P}^*$ must be sampled until there exists a witness p whose confidence box is separated from that of a by at least ε in the appropriate componentwise sense. Since the algorithm uses ℓ_∞-widths, a sufficient condition is that the half-widths of the relevant coordinates shrink below a constant fraction of $\Delta_a - \varepsilon$. Under sub-Gaussian noise this requires $O(\sigma_K^2/(\Delta_a - \varepsilon)^2)$ samples per such a, with an additional $\log(nm/\delta)$ factor from uniform confidence.

Two remarks are worth recording. First, the bound is *insensitive* to the size of $\mathcal{P}^*$: if most architectures are far from the front (large Δ_a), then they are eliminated with few or no $k = K$ samples. Second, the dependence on $m = 1 + 3D$ is only logarithmic via the confidence union bound, while the per-iteration computational overhead may still scale with m through dominance checks; we treat this as an orthogonal issue.

5.4 Calibration for unknown β_k and unknown ρ

The preceding theorems assume that β_k and ρ are known (or conservative upper bounds are available). In deployments, one may only have approximate bias/drift information. We briefly describe an extension that replaces β_k and ρ by data-driven upper bounds obtained from calibration tests; the main algorithm is unchanged except for using these calibrated radii.

Estimating fidelity bias radii. Assume we can evaluate a small calibration subset $\mathcal{C} \subseteq \mathcal{A}$ at two fidelities k and K sufficiently many times in close temporal proximity so that drift is negligible relative to sampling error (or, more conservatively, included in the bound). For each $a \in \mathcal{C}$, consider the empirical difference

$$\widehat{b}_k(a) := \overline{Y}_{a,k}(t) - \overline{Y}_{a,K}(t),$$

where the two means are computed from paired repetitions. A union bound over $a \in \mathcal{C}$ and coordinates $j \in [m]$ yields, with probability at least $1 - \delta_{\mathrm{cal}}$,

$$\|b_k(a)\|_\infty \le \|\widehat{b}_k(a)\|_\infty + \mathrm{rad}_k(a),$$

for an explicit $\mathrm{rad}_k(a)$ determined by σ_k, σ_K and the number of paired samples. We may then define

$$\widehat{\beta}_k := \max_{a \in \mathcal{C}} \big(\|\widehat{b}_k(a)\|_\infty + \mathrm{rad}_k(a)\big),$$

and run MF-ParetoLUCB using $\widehat{\beta}_k$ in place of β_k. This yields the same form of correctness guarantee with δ replaced by $\delta - \delta_{\mathrm{cal}}$, provided $\widehat{\beta}_k$ upper bounds the true bias radii on the confidence event.

Estimating drift. Similarly, suppose we repeatedly measure a fixed anchor set $\mathcal{A}_{\text{anc}} \subseteq \mathcal{A}$ at $k = K$ across time. For each anchor a we may define an empirical maximal deviation across measurement times (after subtracting empirical means) and inflate it by a noise radius to obtain $\widehat{\rho}$ such that

$$\|f_t(a) - f_{t'}(a)\|_\infty \le 2\widehat{\rho} \quad \text{for all observed } t, t'$$

with high probability. Interpreting f as a reference midpoint then yields $\|f_t(a) - f(a)\|_\infty \le \widehat{\rho}$ on the same event. Using $\widehat{\rho}$ in place of ρ in MF-ParetoLUCB preserves Theorem 5.2 with the drift term replaced accordingly.

We emphasize that these calibration steps do not require modeling the relationship between fidelities or the time evolution of drift; they only require repeated measurements and conservative concentration bounds. The cost of calibration can be charged to the same total budget, or treated as a one-time amortized expense if the hardware environment is stable over multiple runs.

In summary, MF-ParetoLUCB admits high-probability correctness guarantees in the presence of both fidelity bias and bounded drift, achieves instance-dependent high-fidelity complexity controlled by Pareto gaps, and extends naturally to settings where bias and drift must be upper bounded from data via calibration. The next section shows that these rates are essentially unimprovable in the worst case by establishing matching lower bounds up to constants and logarithmic factors.

5.5 Matching lower bounds and near-tightness

We complement the upper bounds by showing that, absent additional structure beyond a finite set $\mathcal{A}$ and noisy oracle access, the dependence on the Pareto gaps $\{\Delta_a\}$ in Theorem 5.3 is information-theoretically necessary. For clarity, we state lower bounds first in the unbiased highest-fidelity regime ($k = K$, $\beta_K = 0$) and without drift ($\rho = 0$), since any lower bound in this simpler setting also applies *a fortiori* to the multi-fidelity setting with bias and to environments with temporal variability. We then explain how the multi-objective nature of the problem yields essentially the same change-of-measure obstacles as classical best-arm identification, and finally we discuss when one can beat the finite-set minimax rate by exploiting additional geometric or parametric structure.

From Pareto identification to hypothesis testing. Fix an algorithm Alg that adaptively queries architectures and returns a set $\widehat{\mathcal{P}} \subseteq \mathcal{A}$. Consider two instances $\mathcal{I}$ and $\mathcal{I}'$ that differ only in the true objective vector of a single architecture a, with all other architectures unchanged. If, under $\mathcal{I}$, the architecture a is (ε)-dominated (and hence should be excluded from any certified ε-Pareto approximation), while under $\mathcal{I}'$ the same a becomes ε-nondominated (and hence must be represented, directly or by an ε-close

surrogate), then any (ε, δ)-correct algorithm must be able to distinguish $\mathcal{I}$ from $\mathcal{I}'$ with error probability at most δ. This is a sequential testing problem along the algorithm's adaptive transcript. Standard arguments imply that if Alg does not query a sufficiently often, then the induced transcript distributions under $\mathcal{I}$ and $\mathcal{I}'$ remain too close in KL divergence to support such a distinction.

To formalize this, let N_a be the (random) number of highest-fidelity queries of architecture a, and let $\mathbb{P}_{\mathcal{I}}$ denote the law of the entire transcript (actions and observations) under instance $\mathcal{I}$. For sub-Gaussian (in particular Gaussian) noise models with coordinatewise proxy variance σ_K^2, shifting the mean of $Y_K(a)$ by a vector $u \in \mathbb{R}^m$ yields a per-sample KL divergence on the order of $\|u\|_2^2/\sigma_K^2$ (up to absolute constants, and with the appropriate norm if one works coordinatewise). By the chain rule for KL divergence and optional stopping, one obtains inequalities of the schematic form

$$\mathrm{KL}\big(\mathbb{P}_{\mathcal{I}}, \mathbb{P}_{\mathcal{I}'}\big) \;\le\; c_0\, \mathbb{E}_{\mathcal{I}}[N_a] \cdot \frac{\|u\|_2^2}{\sigma_K^2}, \tag{1}$$

where $c_0 > 0$ is a universal constant. On the other hand, Le Cam's or Bretagnolle–Huber's inequality lower bounds the KL divergence needed to make a decision with error at most δ: for any event E measurable with respect to the transcript,

$$\mathbb{P}_{\mathcal{I}}(E) + \mathbb{P}_{\mathcal{I}'}(E^c) \;\ge\; \frac{1}{2}\exp\big(-\mathrm{KL}(\mathbb{P}_{\mathcal{I}}, \mathbb{P}_{\mathcal{I}'})\big),$$

so that driving both errors below δ forces $\mathrm{KL}(\mathbb{P}_{\mathcal{I}}, \mathbb{P}_{\mathcal{I}'}) \gtrsim \log(1/\delta)$. Combining with (1) yields $\mathbb{E}[N_a] \gtrsim \sigma_K^2 \log(1/\delta)/\|u\|_2^2$. The remaining task is to construct, for each $a \notin \mathcal{P}^*$, a perturbation u small enough to make the test hard, yet large enough to flip (approximate) Pareto status.

Instance-dependent lower bound in terms of Pareto gaps. Let $a \notin \mathcal{P}^*$ and recall that Δ_a quantifies how close a is to the Pareto front in the componentwise sense relevant to elimination. Intuitively, if Δ_a is small, then there exists some $p \in \mathcal{P}^*$ that nearly dominates a; hence a perturbation of size Δ_a to a subset of a's coordinates can remove this domination, turning a into a (nearly) nondominated point. This creates a local indistinguishability region of radius Δ_a around the true mean vector $f(a)$.

A canonical construction is as follows. Choose a witness $p_a \in \mathcal{P}^*$ that attains (or nearly attains) the definition of Δ_a. Modify only $f(a)$ by subtracting a vector $u \ge 0$ (componentwise) supported on the coordinates on which p_a is strictly better than a, with $\|u\|_\infty \asymp \Delta_a - \varepsilon$. Under the modified instance $\mathcal{I}'$, the architecture a becomes ε-nondominated: no single competitor can ε-dominate it because we have reduced exactly the coordinates that previously certified domination. Because the oracle is unbiased at $k = K$,

only queries to a can reveal the shift, and the sub-Gaussian noise masks shifts of size $\|u\|_\infty$ unless N_a is large enough.

Specializing the KL computation to a coordinatewise shift u of magnitude $\Theta(\Delta_a - \varepsilon)$ yields a lower bound of the form

$$\mathbb{E}[N_a] \;\geq\; c_1 \, \frac{\sigma_K^2}{(\Delta_a - \varepsilon)_+^2} \log \frac{1}{\delta},$$

for a universal $c_1 > 0$, where $(x)_+ = \max\{x, 0\}$. Summing over $a \notin \mathcal{P}^*$ gives the instance-dependent lower bound reported (up to constants) in Theorem 4. In particular, this shows that the scaling $\sigma_K^2/(\Delta_a - \varepsilon)^2$ cannot be improved in general, even if one ignores computational cost and allows arbitrary adaptive sampling.

Reduction from (multi-objective) best-arm identification. The preceding argument can be viewed as a direct multi-objective analogue of standard pure-exploration lower bounds, but it is also instructive to see an explicit embedding of single-objective best-arm identification. Consider a classical instance with n arms and unknown means $\mu(a) \in \mathbb{R}$, where the goal is to identify an ε-optimal arm. Embed it into our Pareto problem by defining m objectives with

$$f_1(a) := \mu(a), \qquad f_j(a) := C \;\; \text{for } j = 2, \ldots, m,$$

for an arbitrary constant C. Then componentwise dominance reduces to ordering in the first coordinate, and $\mathcal{P}^*$ consists of the best arm(s). Any algorithm that outputs an ε-Pareto approximation in the multi-objective sense therefore solves best-arm identification with the same confidence. Consequently, the known lower bounds for best-arm identification (gap-dependent $\sum_{a \neq a^*} \sigma^2/\Delta_a^2 \log(1/\delta)$) immediately transfer. This reduction demonstrates that, even though we consider a Pareto set rather than a single best arm, the fundamental difficulty of distinguishing close means under noise is already present in one coordinate, and thus cannot be circumvented by multi-objective reasoning alone.

Minimax lower bound without gap assumptions. The gap-dependent lower bound is tight when the instance has well-separated dominated points. In the absence of any such separation, however, one obtains a minimax lower bound driven by the possibility that *every* architecture could lie on (or arbitrarily close to) the Pareto front. Concretely, for any algorithm and any fixed sampling budget, one can construct an instance where the objective vectors are mutually incomparable (each architecture trades off against the others), and then perturb each $f(a)$ within an ε-scale neighborhood to switch whether it is ε-nondominated. Preventing such switches with probability $1 - \delta$ forces at least constant-order information about each a, which in turn implies that

the number of highest-fidelity evaluations must scale at least linearly in n (and, in quantitative forms, at least on the order of $n\sigma_K^2\varepsilon^{-2}\log(1/\delta)$ under standard noise models). This formalizes the informal statement that, for an unstructured finite set, no algorithm can guarantee sublinear-in-n exploration in the worst case, because the Pareto front may contain a constant fraction of the candidates.

Near-tightness of our upper bounds. Comparing Theorem 5.3 with the instance-dependent lower bound shows that MF-ParetoLUCB achieves the correct $\sum \sigma_K^2/(\Delta_a - \varepsilon)^2$ scaling up to universal constants and logarithmic factors. The remaining discrepancy is the extra $\log(nm)$ factor arising from uniform confidence over all architectures and all objectives. This term is standard in LUCB-style analyses; removing it typically requires either (i) sharper time-uniform concentration combined with data-dependent stopping boundaries, or (ii) a refined change-of-measure analysis that tracks only the subset of arms and coordinates that are *actually* sampled often. We do not pursue these refinements here, since our primary objective is to characterize how multi-objective elimination, multi-fidelity bias control, and drift interact, and the dominant dependence on $(\Delta_a - \varepsilon)^{-2}$ is already optimal.

When can one do better? The lower bounds above are worst-case over *unstructured* finite sets. If additional structure links the architectures, then the relevant complexity measure can change from n (or $\sum_a 1/\Delta_a^2$) to an intrinsic dimension or covering number. One example is a descriptor map $x : \mathcal{A} \to \mathbb{R}^p$ and a Lipschitz condition $\|f(a) - f(a')\|_\infty \le L\|x(a) - x(a')\|$. In such settings, it can be possible to infer bounds on many architectures from measurements on a carefully chosen subset, yielding rates that depend on the metric entropy of $\{x(a)\}$ rather than on n. Another example is a parametric model $f(a) = g(x(a); \theta)$ with low-dimensional θ, where active learning can focus sampling on informative architectures and certify the Pareto set by propagating confidence through the model. These improvements are not available under our standing assumptions, and any such gains must be paid for by additional modeling assumptions whose validity must be checked empirically.

In summary, the gap-dependent upper bound for the number of highest-fidelity queries is essentially optimal for the finite-set, noise-only oracle model, and the linear-in-n minimax barrier explains why multi-fidelity evaluations and elimination are indispensable in practice: without structure, the only way to reduce expensive $k = K$ usage is to exploit that most architectures are separated from the Pareto front and can be discarded early based on coarse, possibly biased measurements.

5.6 Instrumentation and measurement protocols

We specify a measurement protocol intended to (i) produce consistent estimates of per-device cost coordinates $(\tau_d(a), e_d(a), \mu_d(a))$, (ii) support coordinatewise confidence bounds compatible with our sub-Gaussian noise abstraction, and (iii) detect and mitigate temporal drift and toolchain-induced variability. Throughout, we treat each device $d \in \mathcal{D}$ as defining its own measurement harness, but we enforce identical semantic definitions of "token," "generated length," and "peak memory" across devices to preserve comparability of the objective vector $f(a)$.

Canonical inference workload and token accounting. All latency/energy/memory measurements are taken on a fixed inference workload specified by a tuple $(\mathsf{tok}, P, G, \mathsf{dec})$, where tok is the tokenizer/version, P is the prompt length in tokens, G is the number of generated tokens, and dec specifies decoding determinism (e.g., greedy with temperature 0, fixed seed, no sampling). We fix batch size 1 unless explicitly studying throughput; in particular, $\tau_d(a)$ is intended as latency per generated token for interactive use. We log exact token counts as returned by tok including special tokens, and we report all per-token quantities normalized by the realized number of generated tokens G (not by characters or bytes). For decoder-only models we separate the *prefill* phase (processing the prompt) from the *decode* phase (generating tokens with KV cache), since they exhibit different scaling; our primary coordinate $\tau_d(a)$ is decode latency per token, while we additionally record the prefill latency $\tau_d^{\text{prefill}}(a)$ as an auxiliary diagnostic (not part of $f(a)$ unless stated). Concretely, if $T_d^{\text{decode}}(a)$ is wall-clock time elapsed during token generation for G tokens, we define

$$\tau_d(a) \;:=\; \frac{T_d^{\text{decode}}(a)}{G} \quad \text{(ms/token)}.$$

We report time in milliseconds with a monotone clock, and we include framework-side overheads that affect end-to-end user latency (kernel launches, synchronization, and device-to-host transfers required for decoding), but we exclude one-time setup costs such as model loading and compilation (handled separately below).

Latency measurement and synchronization rules. On each device, we implement a micro-benchmark harness that (i) pins the model to the target device, (ii) performs a warm-up stage to amortize transient compilation/caching effects, and (iii) executes repeated timed trials. Warm-up consists of at least W full inference runs on the canonical workload, where W is chosen so that subsequent runtimes stabilize under a simple stationarity check (e.g., the median of the last $W/2$ runs differs from the median of the first $W/2$ runs by at most a small threshold). Timed trials use

explicit device synchronization boundaries to ensure that measured intervals correspond to actual device execution time (e.g., CUDA events with `torch.cuda.synchronize()` or platform-native equivalents). Each timed trial records: prompt token count, generated token count, prefill time, decode time, and total time. We summarize decode latency using the sample mean for concentration analysis and also record the sample median for robustness reporting.

To reduce toolchain variability, we fix: framework version, compiler flags, kernel selection settings, and numerical precision mode (e.g., FP16/BF16/INT8). When compilation is used (e.g., XLA, `torch.compile`, TensorRT), we separate (a) compilation time as a distinct metric and (b) steady-state inference time after compilation, and we ensure that timed trials only begin after compilation completes and caches are populated. When the device supports fixed-frequency modes, we place the device in a stable performance state (e.g., disable CPU turbo boost, set GPU application clocks where permitted, select "performance" governor), and we document any settings that cannot be controlled due to permissions.

Energy measurement and per-token normalization. Energy per token is defined as total incremental energy consumed during decode divided by G:

$$e_d(a) \;:=\; \frac{E_d^{\mathrm{decode}}(a)}{G} \quad \text{(J/token)}.$$

We measure incremental energy $E_d^{\mathrm{decode}}(a)$ using the best available on-device sensor or external meter for each platform, preferring direct energy counters when available (e.g., RAPL on x86, vendor energy counters on mobile SoCs, or GPU energy APIs), and otherwise integrating power samples. In the power-integration case, if power is sampled at times $t_0 < \cdots < t_s$ with readings $P(t_i)$ (watts), we compute energy by trapezoidal integration over the decode window $[t_{\mathrm{start}}, t_{\mathrm{end}}]$:

$$E_d^{\mathrm{decode}}(a) \;\approx \sum_{i:\,[t_i,t_{i+1}]\subseteq[t_{\mathrm{start}},t_{\mathrm{end}}]} \frac{P(t_i)+P(t_{i+1})}{2}\,(t_{i+1}-t_i).$$

We align the decode window to the same synchronization boundaries as latency measurement. If the energy interface reports total device energy (including baseline idle), we subtract an idle baseline measured in a matched window with identical synchronization but no model execution, yielding an incremental estimate. We record sensor sampling rate and quantization; if the sampling rate is too low to resolve per-token dynamics, we lengthen G so that $E_d^{\mathrm{decode}}(a)$ is well above sensor noise, while keeping the same per-token normalization.

Peak memory footprint: definition and measurement. Peak memory $\mu_d(a)$ is defined as the maximum resident memory attributable to the model execution during the canonical workload, including weights, KV cache, and framework allocations required for decoding. Because memory accounting differs across platforms, we enforce a semantic rule: $\mu_d(a)$ is the peak *device memory* used by the process (or container) running inference. On GPUs we use the framework-reported maximum allocated memory and/or driver-reported peak usage; on CPU we use peak RSS; on mobile we use platform APIs for peak resident memory when available. To mitigate fragmentation effects, we perform measurements in a fresh process where feasible, and we distinguish *static* memory (model weights and persistent buffers) from *dynamic* memory (KV cache growth with $P + G$). We therefore log a memory trace over time and record both the peak and its timing relative to decode steps. The reported $\mu_d(a)$ is the peak observed over timed trials after warm-up; if trials vary due to allocator behavior, we treat this as noise and incorporate it into confidence bounds.

Repetitions, aggregation, and confidence intervals. For each architecture a and device d, we collect R repeated timed trials at a given fidelity k, yielding observations $Y_{k,r}(a) \in \mathbb{R}^m$ whose relevant coordinates are computed as above. We aggregate per-coordinate using the empirical mean $\widehat{f}_j(a)$ (after the appropriate mapping from raw timings/energies/memory to per-token quantities). For uncertainty, we use coordinatewise confidence radii compatible with sub-Gaussian tails. Concretely, if coordinate j at fidelity k has proxy variance bound σ_k^2, then for confidence level α we may take

$$r_{a,j}(R, \alpha) \;:=\; \sqrt{\frac{2\sigma_k^2 \log(2/\alpha)}{R}},$$

and we allocate α via a union bound across (a, j) and across updates as required by the algorithmic analysis. When σ_k^2 is not known a priori, we estimate it conservatively from pilot measurements and inflate by a safety factor; we additionally verify empirically that residuals are light-tailed (e.g., by checking that extreme deviations are consistent with the chosen proxy). We emphasize that our algorithmic guarantees rely on conservative upper bounds rather than exact parametric correctness.

Change-point detection and drift handling. We operationalize drift as time-variation in effective objectives $f_t(a)$ induced by background load, thermal effects, firmware updates, or measurement stack changes. We implement two complementary mechanisms. First, we maintain a *sentinel set* $\mathcal{S} \subseteq \mathcal{A}$ of fixed architectures that are periodically re-measured on each device; these sentinels are chosen to span typical runtime/memory regimes (small/medium/large). For each sentinel $s \in \mathcal{S}$ and each coordinate j, we

track a time series of residuals relative to the current reference estimate. We apply a simple sequential change-point test (e.g., Page–Hinkley or CUSUM on standardized residuals) to flag abrupt shifts. Second, we estimate a drift bound ρ by taking the maximum absolute deviation observed between two measurements of the same (s, j) within a prescribed time window, after subtracting estimated measurement noise; this yields a conservative ℓ_∞ drift envelope that can be fed into the confidence inflation used by MF-ParetoLUCB.

Upon detecting a change-point on device d, we take one of two actions depending on severity: (i) *recalibration*, in which we temporarily pause exploration and re-measure $\mathcal{S}$ to update the drift bound and re-anchor confidence intervals; or (ii) *segmentation*, in which we treat measurements before and after the change as belonging to different regimes and avoid mixing them in a single estimator. In either case, we log the event and annotate the transcript so that subsequent analysis can attribute variance either to stochastic noise or to systematic drift.

Toolchain variability and reproducibility controls. To ensure that "noise" is not dominated by uncontrolled software variability, we record a full toolchain manifest per device: OS/kernel build, driver versions, framework/library versions, model serialization format, quantization/calibration artifacts, and compiler cache identifiers. We pin CPU affinity and isolate benchmarking processes where possible, disable extraneous background services, and enforce deterministic decoding settings. When such controls are infeasible (e.g., shared mobile devices), we treat the resulting variability as part of the measurement noise and increase repetition counts R accordingly. We also ensure that any fidelity-k proxy (e.g., simulator, reduced sequence length, smaller batch) uses the *same* tokenization and reports per-token metrics under the same definitions, so that bias $b_k(a)$ reflects approximation error rather than definitional mismatch.

Consistency checks and derived quantities. Finally, we implement cross-metric sanity checks: latency and energy must be nonnegative; energy must be broadly consistent with average power bounds for the device; and peak memory must exceed static weight size up to compression factors. We also verify scaling with G: decode time should be approximately linear in G once warm-up completes, and memory growth should match KV cache dimensionality. When a measurement violates these checks (e.g., due to thermal throttling mid-run), we retain the data but annotate it and, if necessary, trigger drift handling. This protocol yields a coherent set of per-token and peak-memory coordinates suitable for the oracle model $Y_k(a) = f(a) + b_k(a) + \xi$ and supports the construction of conservative confidence bounds used by the elimination and certification logic.

5.7 Experimental plan (benchmarks, baselines, metrics, ablations)

We outline an experimental plan to evaluate MF-ParetoLUCB as a procedure for identifying a certified approximate Pareto set under a costed multi-fidelity oracle, with objectives given by $f(a) = (\ell(a), (\tau_d(a), e_d(a), \mu_d(a))_{d\in\mathcal{D}})$. Our primary goal is not to exhaustively optimize any particular application, but rather to stress-test (i) correctness of elimination/certification under bias and drift, (ii) budget efficiency in terms of high-fidelity evaluations, and (iii) robustness of the returned set $\widehat{\mathcal{P}}$ under reruns.

Architecture search spaces. We consider finite candidate sets $\mathcal{A}$ constructed by enumerating and filtering discrete design choices, with sizes ranging from $n \approx 10^2$ (for near-exhaustive diagnostics) to $n \approx 10^4$ (for budget-limited regimes). We propose three families.

1. *Decoder-only Transformer variants.* We enumerate tuples $(L, d_{\text{model}}, d_{\text{ff}}, h, \mathsf{attn}, \mathsf{kv}, \mathsf{prec})$ where L is layer count, h heads, and prec is an inference precision/quantization mode. We include attention/kernel choices attn (standard, fused, or sparse variants) and KV-cache format choices kv (e.g., standard vs. compressed). Each tuple maps to a concrete implementation artifact, so each $a \in \mathcal{A}$ has a well-defined $f(a)$ in the sense of our oracle model.

2. *Operator-set ablations.* We construct a finite set by toggling a small number of implementation operators that materially affect device costs (e.g., activation variants, layernorm placement, tensor-parallel degree on GPU, and memory-saving options), holding the high-level parameter count approximately fixed. This isolates the multi-device cost structure while keeping quality changes modest but non-negligible.

3. *Compression/serving configurations.* For a fixed pretrained backbone, we enumerate discrete serving-time configurations: quantization level, pruning/sparsity setting, and batch/sequence constraints. This produces a large $\mathcal{A}$ with relatively tight quality range but wide per-device cost variation.

In all cases we record a deterministic mapping from the symbolic architecture description to an executable artifact to ensure that repeated queries of the same (a, k) correspond to the same underlying implementation.

Task suites and the quality objective. We define $\ell(a)$ as an aggregate task loss (to be minimized) on a fixed validation suite. Concretely, we propose to use a mixture of (i) language modeling loss on held-out text, (ii) instruction-following or summarization loss under teacher forcing when applicable, and (iii) a small code or reasoning subset, with task weights fixed

a priori. When accuracy-style metrics are more natural, we convert them to minimization form (e.g., $\ell(a) = -\mathrm{Acc}(a)$) and, when necessary, rescale to comparable numeric ranges to avoid hypervolume calculations being dominated by a single coordinate. We emphasize that MF-ParetoLUCB itself is invariant to monotone coordinate transforms, but downstream evaluation metrics (notably hypervolume) are not.

Devices and objective dimensionality. We choose a heterogeneous device set $\mathcal{D}$ spanning at least one server-class GPU, one datacenter CPU, and one edge-class device (e.g., mobile GPU/NPU). This yields $m = 1 + 3D$ objectives, and we explicitly evaluate the regime in which m is moderately large (e.g., $D \in \{3, 4, 5\}$), since multi-device Pareto structure is the motivating application. All per-device cost coordinates are measured following the protocol of Section 5.6; here we focus only on how those measurements enter the experimental comparison.

Multi-fidelity ladder design. We instantiate a ladder $k \in \{1, \ldots, K\}$ with $K \in \{2, 3\}$, choosing fidelities that are operationally meaningful and that plausibly satisfy $\|b_k(a)\|_\infty \leq \beta_k$ with conservative, auditable β_k. A representative $K = 3$ design is:

1. $k = 1$: a cheap proxy (e.g., an analytic estimator, a learned predictor trained on past runs, or a simulator) returning a full vector estimate of $f(a)$, with a large but known β_1.

2. $k = 2$: a partial real measurement (e.g., shorter generation length, reduced repetition count, or profiling-only without full-quality evaluation), with intermediate β_2 and reduced cost c_2.

3. $k = 3 = K$: full protocol measurement with $\beta_K = 0$.

We pre-register costs c_k in a common unit (e.g., wall-clock time converted to a monetary or GPU-hour equivalent) and run MF-ParetoLUCB under a fixed total budget B. In addition to fixed-budget runs, we include a fixed-accuracy mode in which we terminate only under the stopping rule and treat budget overruns as failures (reported explicitly).

Baselines. We compare against methods chosen to isolate the contribution of (i) multi-objective elimination with confidence bounds, (ii) multi-fidelity bias management, and (iii) adaptive focus on Pareto-ambiguous candidates.

1. *Random (high-fidelity only).* Sample architectures uniformly from $\mathcal{A}$, evaluate at $k = K$ until budget is exhausted, and return the nondominated subset among evaluated points. This baseline clarifies the benefit of adaptive sampling versus passive collection.

2. *Predictor-only Pareto selection.* Fit a surrogate $\widetilde{f}(a)$ from an initial design (either random or space-filling), then select a predicted Pareto set and spend the remaining budget verifying only those candidates at $k = K$. This tests whether MF-ParetoLUCB is doing more than "predict then verify."

3. *Random + weight-sharing / one-shot proxy.* Train a shared model (where applicable) or use a one-shot estimator to cheaply rank candidates; then evaluate a small set at $k = K$. We include this because it is common in NAS practice and creates an implicit fidelity with potentially nontrivial bias.

4. *Multi-objective NAS baselines (e.g., MONAS/DPP-style selection).* We instantiate a multi-objective search method that proposes a batch of candidates each round (e.g., using diversity-promoting selection or evolutionary updates) under the same budget accounting. Where such methods lack explicit certification, we still evaluate their returned set under the same metrics as below.

5. *Scalarization sweeps.* For a grid of weights $w \in \Delta^{m-1}$, run a single-objective best-arm procedure (or simple bandit elimination) on $\langle w, f(a)\rangle$, then merge all identified candidates and take the nondominated subset. This baseline approximates a common engineering approach to Pareto discovery.

All baselines are run under identical oracle access, i.e., the same fidelities and costs when applicable. If a baseline cannot naturally exploit multi-fidelity, we record this as a design limitation rather than artificially granting it extra information.

Evaluation metrics. Since the output is a set $\widehat{\mathcal{P}} \subseteq \mathcal{A}$, we use set-valued metrics capturing both quality of approximation and stability.

1. *Hypervolume (HV).* After normalizing coordinates to $[0, 1]$ using fixed lower/upper bounds (derived either from known feasible ranges or from a precomputed reference set), we compute dominated hypervolume of $\widehat{\mathcal{P}}$ with respect to a fixed reference point $r \succ \max_a f(a)$ (in normalized minimization coordinates). For larger m, we use a Monte Carlo estimator with a fixed random seed and sufficient samples to make estimator variance negligible relative to inter-method differences.

2. *Empirical Pareto regret.* On instances where we can afford an (approximately) exhaustive high-fidelity evaluation to form a reference front $\mathcal{P}^*$ (or a high-quality proxy thereof), we report the smallest $\eta \geq 0$ such that for every a in the reference set there exists $\hat{p} \in \widehat{\mathcal{P}}$ with $f(\hat{p}) \preceq f(a) + \eta\mathbf{1}$. This directly aligns with our additive approximation notion and yields an interpretable scalar.

3. *Certification and contamination.* We report (i) the fraction of $\widehat{\mathcal{P}}$ that is truly nondominated up to a small tolerance under high-fidelity reevaluation and (ii) the fraction of true Pareto points (or reference points) that are covered up to tolerance by $\widehat{\mathcal{P}}$.

4. *Cost decomposition.* We report total cost $\sum_i c_{k_i}$ and the number N_K of highest-fidelity queries, since the central claim is reduction in expensive evaluations.

5. *Stability under reruns.* We rerun each method with different random seeds and (when feasible) different measurement times. We report set similarity (e.g., Jaccard index on the identity of architectures returned) and metric stability (standard deviation of HV and empirical Pareto regret). We additionally report the maximum coordinatewise deviation in reevaluations of returned candidates to quantify susceptibility to drift.

Ablations. To isolate design choices in MF-ParetoLUCB, we include controlled ablations:

1. *No drift handling.* Set $\rho = 0$ and disable any remeasurement/sentinel-driven inflation, while keeping all else fixed. This tests whether the empirical instability matches the theoretical 2ρ penalty.

2. *No multi-fidelity.* Restrict to $k = K$ (thus $\beta_k \equiv 0$), reducing MF-ParetoLUCB to a single-fidelity Pareto elimination scheme. This isolates the value of cheap biased information.

3. *No elimination (uniform allocation).* Continue sampling without discarding certainly dominated points, allocating budget either uniformly over $a \in \mathcal{A}$ or proportional to current uncertainty widths. This isolates the effect of safe elimination on sample complexity.

4. *No bias-aware promotion.* Allow low-fidelity sampling but ignore β_k in confidence inflation and promotion logic. This tests whether explicit bias control is necessary in practice.

Reporting format. We will report curves of HV and empirical Pareto regret versus consumed budget B, as well as N_K versus achieved regret, to expose the cost–accuracy tradeoff. For each benchmark we will include a small-number-of-points visualization of the front on selected coordinate pairs (e.g., ℓ vs. τ_d for each d) while emphasizing that the true object is m-dimensional. All plots and tables will be accompanied by rerun variability summaries to distinguish stochasticity from systematic drift and bias.

5.8 Discussion and limitations

Dependence on a finite candidate set. Our guarantees are stated for a finite architecture set $\mathcal{A}$ and proceed by uniform concentration over $(a, j) \in \mathcal{A} \times [m]$. This is not merely a technical convenience: the elimination logic in MF-ParetoLUCB is defined by explicit dominance comparisons among confidence boxes indexed by $a \in \mathcal{A}$, and Theorem 2 is ultimately a statement about set containment and certificates among these finitely many alternatives. In applications, $\mathcal{A}$ is typically induced by discretizing a design space (layers, widths, operator toggles, quantization modes) and then filtering by feasibility constraints. The quality of the returned $\widehat{\mathcal{P}}$ is therefore limited by the expressivity of this discretization: if the true deployment-optimal design lies between grid points, no algorithm operating on $\mathcal{A}$ can recover it.

The finiteness assumption also interacts with complexity. While the sample complexity statements scale logarithmically in n, the worst-case computational cost of maintaining dominance relations can scale as $\Theta(n^2 m)$ per round without further geometric data structures. Hence, even if oracle evaluations dominate wall-clock time in many regimes, there exist regimes (large n, moderate m, very cheap proxies) where naive dominance maintenance becomes a bottleneck. Our analysis does not attempt to optimize this aspect; rather, it isolates the statistical cost of certification under bias and drift. In practice, one should expect to rely on incremental skyline maintenance, aggressive pruning by cheap feasibility checks, and caching of dominance witnesses.

Choice and meaning of β_k: bounded bias is a strong modeling commitment. The multi-fidelity abstraction $Y_k(a) = f(a) + b_k(a) + \xi$ with $\|b_k(a)\|_\infty \le \beta_k$ makes two commitments: (i) bias is bounded uniformly over $\mathcal{A}$, and (ii) the bound β_k is known to the algorithm. Both can be difficult to justify for learned predictors or simulators trained on historical data, especially under distribution shift across architecture families. If β_k is under-specified, then the confidence boxes can be systematically miscentered, and the "safe elimination" argument used in Theorem 2 may fail, i.e., a truly Pareto-relevant architecture could be eliminated. If β_k is over-specified, correctness is retained but the algorithm may delay promotion decisions and over-explore, eroding the cost advantages of low-fidelity queries.

Two pragmatic mitigations are natural within our framework. First, β_k can be made coordinatewise (and even device-specific): replacing β_k by $\beta_{k,j}$ is immediate and often more realistic, since latency simulators, memory estimators, and quality proxies can have qualitatively different error scales. Second, β_k can be calibrated empirically by auditing a random subset of architectures at both fidelity k and K, and setting β_k to a high quantile of the observed discrepancies inflated by a statistical tolerance. Such calibra-

tion does not prove a uniform bound over all $\mathcal{A}$, but it converts an implicit heuristic into an explicit, auditable assumption. From a theoretical standpoint, replacing a worst-case β_k by a high-probability bound simply changes the bookkeeping of the overall failure probability via an additional union bound (or a two-stage δ-allocation) without altering the logic of the method.

Drift bound ρ: identifiability versus realism. We include drift through the deterministic bound $\|f_t(a) - f(a)\|_\infty \le \rho$, leading to the unavoidable additive 2ρ penalty (Corollary 5). This is the correct worst-case statement when measurements of different architectures occur at different times and drift may adversarially move objectives in opposite directions. Nevertheless, the interpretation of ρ requires care. Drift in hardware measurement is often structured (thermal saturation, background load, driver state) and can be partially controlled by experimental protocol (warmup, fixed clocks, exclusive mode, repeated interleavings). If one can enforce near-simultaneous paired measurements, or if one can model drift as a stochastic process with mixing properties, then the worst-case 2ρ term may be pessimistic, and sharper guarantees may be possible. Our framework does not exploit such structure; it provides a conservative envelope within which certification remains valid.

A further subtlety is that what we call "drift" may also absorb unmodeled systematic error (e.g., measurement instrumentation bias, nondeterminism in kernels, or operator fusion variability across runs). If these effects are not bounded in ℓ_∞ by a small ρ, then we cannot expect any method to deliver reliable componentwise certificates without strengthening the observation model (e.g., by explicitly tracking run conditions as covariates). In particular, the assumption $\beta_K = 0$ should be interpreted as "no fidelity bias relative to the chosen reference protocol," not as "perfect truth."

Proxy mismatch and non-monotone fidelities. The ladder assumption β_k nonincreasing in k encodes the idea that higher fidelities are closer to the deployment protocol. In practice, one can encounter non-monotone behavior: a "medium fidelity" measurement (short sequence length, fewer repetitions) may exhibit less bias for some architectures than a longer run, due to cache effects or kernel selection thresholds. Likewise, learned predictors may be accurate on certain operator subsets and poor on others, yielding architecture-dependent bias patterns that violate uniformity.

Our algorithmic prescription in such cases is not to force a dubious monotone ladder, but to refine the fidelity design until the bound is defensible. Concretely, one may maintain multiple proxies as separate fidelity options and treat β_k as a conservative bound for each, allowing the selection rule to choose among them by cost and estimated informativeness. If no meaningful β_k can be certified, then the method degenerates to a single-fidelity scheme

(still useful) and the remaining question becomes one of variance reduction and efficient elimination under noise alone.

Additive accuracy and scaling across objectives. Our correctness notion is additive: $\widehat{\mathcal{P}}$ $(\varepsilon + 2\rho)$-covers $\mathcal{A}$ in the componentwise order. This is mathematically natural under bounded ℓ_∞ errors, but it depends on the units of the objectives. Latency in ms/token, energy in J/token, and memory in MB have different scales, and even within a single coordinate the scale may change with sequence length or batch size. Hence, ε is only meaningful relative to a declared measurement protocol and a declared normalization. In deployment settings, this is a feature rather than a defect: stakeholders often specify tolerances in concrete units (e.g., "within 0.2 ms/token"), and certification should respect those units. Still, when comparing across tasks or devices, one should explicitly state the normalization and interpret ε accordingly.

Beyond finite $\mathcal{A}$: continuous and evolving search spaces. Extending MF-ParetoLUCB to continuous or combinatorially enormous spaces requires additional structure. One route is discretization with a covering argument: if f is Lipschitz in a continuous parameterization, then an η-net reduces the problem to a finite set with n controlled by the covering number, at the expense of an additional approximation error. Another route is to replace the uniform finite-arm concentration by model-based confidence regions (e.g., linear or kernel bandit assumptions over architecture descriptors), yielding sample complexity in terms of dimension or information gain rather than n. Both directions are viable but fundamentally change the nature of the guarantees: they trade the assumption "$\mathcal{A}$ is finite" for assumptions on regularity or realizability of a surrogate model.

A related practical complication is that operator sets evolve: compilers change, kernels are updated, and new quantization modes appear. In such nonstationary settings, $\mathcal{A}$ may change over time, and even $f(\cdot)$ under the nominal protocol may shift. One can incorporate new candidates by treating the algorithm as operating in a streaming mode with a growing $\mathcal{A}_t$, but then uniform confidence must be re-allocated (or refreshed) over time, and drift control becomes central rather than auxiliary. We view our drift term ρ as a minimal step in this direction, but not a complete solution to evolving software/hardware stacks.

Reproducibility checklist (for auditable certification). Since our claims are conditional on explicit bounds $(\beta_k, \sigma_k, \rho)$ and a well-defined oracle protocol, we recommend reporting the following items whenever MF-ParetoLUCB (or any certified Pareto identification method) is evaluated:

- **Candidate set definition.** Exact specification of $\mathcal{A}$: parameter grids, operator toggles, feasibility filters, and the deterministic mapping from each symbolic a to an executable artifact.
- **Objective protocol.** Precise definition of all m coordinates: datasets/task suite and aggregation for $\ell(a)$; device-specific measurement conditions for $\tau_d(a), e_d(a), \mu_d(a)$ (sequence length, batch size, decoding method, warmup, measurement window).
- **Device and software stack.** Hardware identifiers, OS, drivers, compiler versions, kernel libraries, and any clock/thermal settings; confirmation of exclusive access or description of background load controls.
- **Fidelity ladder.** Operational meaning of each fidelity k, the corresponding cost accounting c_k, and whether fidelity affects quality evaluation, profiling, or both.
- **Noise and repetitions.** Number of repetitions per measurement, estimator used (mean/median), and the method used to upper bound σ_k (or to justify a sub-Gaussian proxy).
- **Bias calibration.** Procedure used to set β_k (and any coordinatewise $\beta_{k,j}$), including the audit set size, statistical confidence level, and any held-out validation of the bound.
- **Drift quantification.** Procedure used to estimate ρ: sentinel architectures, remeasurement frequency, time window, and summary statistics leading to the chosen bound.
- **Randomness control.** Random seeds for algorithmic choices and for any stochastic training/evaluation; reporting of rerun variability.
- **Raw logs.** Storage of per-query outputs $Y_k(a)$, timestamps t, and environment metadata sufficient to recompute $\widehat{f}$, confidence bounds, and the final elimination certificates.

This checklist does not eliminate modeling risk, but it makes the assumptions falsifiable and the results comparable across implementations.

Summary. We have presented MF-ParetoLUCB as a conservative, certificate-driven alternative to heuristic Pareto discovery under multi-device objectives. Its limitations are the limitations of its assumptions: finiteness (or discretization), defensible bias envelopes for proxies, and a credible bound on drift. When these inputs are auditable, the method yields correspondingly auditable outputs; when they are not, one should interpret the procedure as a principled heuristic rather than a correctness-guaranteeing algorithm.

5.9 Conclusion

We have framed *certified Pareto neural architecture search* as a finite-set identification problem in which the object of interest is not a single "best" architecture but a set of mutually non-dominated tradeoffs across quality and deployment costs. The key point is that the output of a NAS procedure is typically consumed as a decision support artifact: practitioners wish to select among alternatives under shifting constraints (latency caps, energy budgets, memory limits, or accuracy targets) and across multiple devices. In such settings, returning a scalarized optimum is brittle, whereas returning a Pareto approximation is aligned with the actual decision surface. Our contribution is to make this output auditable: under explicit assumptions on proxy bias, measurement noise, and bounded drift, we can return a set $\widehat{\mathcal{P}}$ that is $(\varepsilon + 2\rho)$-Pareto accurate with probability at least $1 - \delta$, together with an explicit mechanism (confidence boxes and dominance certificates) that witnesses why candidates are retained or eliminated.

This viewpoint provides a clean bridge from common NAS heuristics to deployment-faithful guarantees. Most NAS pipelines already separate *candidate generation* (evolutionary search, differentiable relaxations, predictor-guided exploration, or manual templates) from *candidate evaluation* (training and profiling). Our formulation isolates the second stage: given a finite candidate set $\mathcal{A}$, the task is to spend a limited evaluation budget to certify which candidates are needed to cover the remaining ones up to an additive tolerance. The algorithm MF-ParetoLUCB does not prescribe how $\mathcal{A}$ is obtained, and in this sense it is compatible with essentially any upstream heuristic; it can be viewed as a wrapper that turns a heuristic proposal set into a set with probabilistic correctness guarantees relative to the declared oracle protocol.

The technical content of the bridge is the explicit accounting for the two failure modes that dominate deployment measurement practice: (i) low-cost proxies are biased relative to the deployment protocol, and (ii) even high-fidelity measurements are noisy and subject to time variation. The oracle model $Y_k(a) = f(a) + b_k(a) + \xi$ with $\|b_k(a)\|_\infty \le \beta_k$ and sub-Gaussian ξ makes bias and noise separable and thus separately controllable by fidelity promotion and replication. The drift model $\|f_t(a) - f(a)\|_\infty \le \rho$ makes explicit that comparisons across time necessarily incur slack, and that any guarantee stated in terms of the static target f must tolerate a 2ρ discrepancy when two architectures are measured at different times. In our analysis, these quantities appear transparently as additive inflations of confidence bounds, so that the algorithm cannot "hide" optimism behind unmodeled variability: it must pay for certainty either by promoting fidelity (reducing β_k) or by sampling (reducing the stochastic radius), and it must accept the irreducible drift penalty unless stronger synchronization is assumed.

From an algorithmic standpoint, MF-ParetoLUCB is a conservative elim-

ination scheme operating on vector-valued confidence boxes. At any time we maintain high-probability bounds $\mathrm{LCB}(a)$ and $\mathrm{UCB}(a)$ for each $a \in \mathcal{A}$, inflated by the current fidelity bias radius and the drift bound. Elimination is then performed only when a dominance relation holds at the level of these boxes with margin ε, which yields a certificate that survives the translation from estimated to true objectives. The output is the set of candidates that cannot be ruled out as Pareto-relevant given the current uncertainty; equivalently, it is the set that still intersects the ε-expanded nondominated region under all plausible realizations consistent with the observations and bounds. This is the precise sense in which the method turns heuristic exploration into certified conclusions: every exclusion is justified by an explicit, checkable inequality involving only stored statistics and declared tolerances.

Our complexity results clarify when expensive deployment-faithful evaluations are actually necessary. The gap-dependent upper bound shows that, after fidelity promotion has reduced proxy bias below the relevant margins, the number of highest-fidelity evaluations required to resolve Pareto status scales as $\sum_{a \notin \mathcal{P}^*} \sigma_K^2/(\Delta_a - \varepsilon)_+^2$ up to logarithmic factors, which matches an instance-dependent lower bound up to constants and $\log(nm)$ terms. Thus the algorithm is not merely correct; it is near-optimal in how it spends high-fidelity budget on hard-to-separate candidates near the Pareto boundary. This near-tightness is important in practice because the cost ratio between fidelities can be extreme: if one must ultimately measure on real devices, we wish to know that these measurements are information-theoretically justified rather than artifacts of an overly cautious procedure.

The multi-objective formulation also makes explicit an often implicit deployment reality: different objectives may be learned or measured with different difficulties, and the bottleneck is the coordinate that blocks dominance resolution. Since $m = 1 + 3D$ can be moderately large, one might worry that certification becomes hopelessly expensive. Our results indicate instead that the essential difficulty is concentrated near the front: if a candidate is clearly dominated in at least one coordinate by a large margin, it is eliminated quickly; if it is near-nondominated, then it must be sampled, and no method can avoid this. This is a useful diagnostic principle: when certification appears expensive, it reflects genuine ambiguity in the tradeoffs at the declared tolerances, not merely inefficiency. Conversely, if certification is cheap, then the front is well-separated at scale ε, and one can trust that the returned set is not an artifact of under-sampling.

In terms of methodology, we emphasize that certification is complementary to, not a replacement for, search. For large design spaces, one may first use aggressive heuristics and cheap proxies to propose a manageable $\mathcal{A}$, and only then invoke certified Pareto identification as a final stage that (i) allocates expensive measurements adaptively, (ii) provides a clear stopping rule tied to (ε, δ), and (iii) outputs a set equipped with guarantees under explicit modeling assumptions. This division of labor is natural: the upstream

heuristic is judged by its ability to generate a candidate set containing good solutions, while the certification stage is judged by its ability to decide, at controlled cost, which of these solutions are needed to cover the rest under the true deployment protocol. When combined, the overall pipeline inherits the practical strengths of heuristic exploration while gaining an explicit correctness layer at the end.

Finally, the central conceptual message is that NAS for deployment should be treated as a problem of *controlled approximation under measurement constraints*. The parameters ε, δ, β_k, σ_k, and ρ are not nuisances; they are the quantities that encode what it means for an empirical claim about tradeoffs to be reliable. By stating them explicitly, we make the promise of the algorithm commensurate with what can actually be measured. The resulting guarantee—an $(\varepsilon+2\rho)$-Pareto approximation with high probability—is modest in form but strong in implication: it asserts that, within declared tolerances and under auditable assumptions, the returned set is sufficient for any downstream selection rule that respects componentwise costs.

We therefore view certified Pareto NAS not as an alternative objective to heuristic NAS, but as a principled interface between empirical search and deployment decision-making. It provides a language in which proxy evaluations, device measurements, and uncertainty can be combined into statements of the form "no excluded architecture can improve any objective without paying at least ε (up to drift)." Such statements are the appropriate unit of scientific and engineering communication for deployment: they are falsifiable, tied to a protocol, and stable under re-interpretation of preferences over objectives. Within these constraints, MF-ParetoLUCB offers a concrete instantiation of the broader idea that the output of NAS should be not only performant, but also certifiably so relative to the system in which it will be used.