Iso-Cost Micro-Mutations for Hardware-Aware NAS: Fair Local Search under Latency/Energy Constraints

Liz Lemma Future Detective

January 20, 2026

Abstract

Modern NAS increasingly targets deployment constraints (latency, energy, activation memory), yet common micro-search decisions (e.g., switching to heavier operators or larger kernels) are confounded by hidden increases in cost. Building on the macro–micro global NAS paradigm and the parameter-normalization idea in Siddiqui et al. (2025), we generalize "parameter-equalized" micro-search to an iso-cost framework for arbitrary hardware metrics. We formalize iso-cost neighborhoods under a target device h, define an iso-cost micro-mutation operator that compensates width/precision to preserve cost within tolerance, and propose a layerwise micro-search algorithm with cost-aware confidence bounds. Under mild assumptions (monotone cost in width/precision, bounded cost-model error, sub-Gaussian proxy noise), we prove the algorithm finds an (ε, τ)-approximate local optimum with $\tilde{O}(Lm\sigma^2/\varepsilon^2)$ proxy evaluations, and show a matching information-theoretic lower bound. We outline how differentiable cost models (or calibrated profilers) on multiple hardware targets enable practical iso-latency/iso-energy comparisons, yielding cleaner Pareto fronts than parameter-based normalization. Empirical validation (recommended) would demonstrate improved ranking reliability and transfer across devices for CNN/ResNet-like and hybrid Conv–Attention spaces.

Table of Contents

4. 4. Problem formulation: (i) iso-cost mutation feasibility, (ii) iso-cost local improvement, and (iii) iso-cost Pareto local search (optional extension).

5. 5. Cost modeling and calibration: defining g_h; estimator $\widehat{g}_h$; measurement protocols; assumptions and how to test/validate η empirically.

6. 6. Iso-cost micro-mutation operator: construction, feasibility conditions, and guarantees that true cost remains within tolerance despite model error.

7. 7. Algorithm: layerwise iso-cost micro-search with confidence bounds; invariants; stopping criteria; optional multi-device variant.

8. 8. Theory: sample complexity upper bound for (ε, τ)-local optimality; matching lower bound; discussion of tightness and assumptions.

9. 9. Practical evaluation plan (implementation-dependent): benchmarks, devices, cost metrics, baselines (parameter-normalization, cost-penalty search), and metrics (Pareto front quality, rank correlation, transfer).

10. 10. Limitations and extensions: non-monotone costs, kernel-level implementation variance, multi-objective iso-cost sets, coupling with dynamic training budgets (from the source).

11. 11. Conclusion: iso-cost as a fairness primitive for micro-search; implications for deployment-aware NAS.

1 1. Introduction: the confounding problem in micro-search; why parameter-normalization is insufficient in 2026; contributions and claims.

Micro-search procedures for neural architectures are typically presented as “local” improvements: one perturbs a baseline network by changing a kernel, an operator, an attention window, or a normalization rule, and one retains the change if the measured utility increases. On current accelerators this informal description hides a persistent confounder. A micro-change rarely acts in isolation; it almost always changes the deployment cost—latency, energy, or activation memory—and the proxy evaluation protocol is itself sensitive to those cost-induced shifts (e.g., via implicit regularization, batch size constraints, or optimization stability). Consequently, when a candidate outperforms a baseline, it is generally unclear whether we have discovered a better micro-choice or merely spent more compute in disguise. We treat this as an identifiability problem: the “architectural effect” and the “cost effect” are entangled unless cost is controlled.

A common response is to normalize by parameter count or nominal FLOPs. In 2026 this is no longer adequate even as a rough surrogate for deployment cost. First, FLOPs ignore the interaction between operator shape and the hardware stack: the same arithmetic count can map either to a compute-bound fused kernel or to a memory-bound sequence of small kernels, and the latter can be an order of magnitude slower despite identical FLOPs. Second, parameter count is insensitive to activation memory and bandwidth, which dominate for attention-like modules, large feature maps, and high-resolution stages. Third, modern runtimes perform graph-level transformations (fusion, tiling, algorithm selection, quantization-aware lowering) that make cost strongly non-linear in seemingly minor micro-choices. Finally, mixed precision and per-layer width scaling alter not only arithmetic throughput but also tensor-core utilization, cache behavior, and memory traffic, so “same parameters” does not imply “same latency,” and “same FLOPs” does not imply “same energy.” Hence any micro-search that compares candidates at unequal true cost risks selecting changes that simply allocate more capacity along the hardware-relevant axes.

We therefore advocate a different principle: micro-search decisions should be made within iso-cost neighborhoods on a fixed hardware target. Concretely, we restrict comparisons to architectures whose deployment cost lies within a tolerance band around a baseline cost. This yields a controlled experimental setting in which observed utility differences are attributable to micro-choices rather than to budget inflation. The tolerance is necessary because cost is measured through a combination of profilers and predictors, and because discrete compilation effects can cause small discontinuities; nevertheless, keeping candidates within a narrow relative band suffices to remove the dominant degree of freedom.

To operationalize this principle we use compensated micro-mutations. Given a proposed local change (e.g., swapping an operator at one layer), we adjust a set of "knobs"—most naturally per-layer widths and/or per-layer precision—to bring the mutated architecture back into the prescribed cost band. The compensation step is not an afterthought: without it, the neighborhood induced by micro-changes is biased toward cost-increasing moves, since many expressive operators are also more expensive. By explicitly enforcing iso-cost feasibility, we ensure that the neighborhood is symmetric in the sense relevant to local optimality: a move is judged by its utility at essentially fixed cost.

This framing also clarifies the statistical burden of micro-search. Proxy evaluations are noisy: short training runs, partial data, and stochastic optimization introduce variability that can overwhelm small architectural gains. Under a fixed proxy protocol, selecting among a small set of iso-cost candidates is a best-arm identification problem with sub-Gaussian noise, and thus admits fixed-confidence procedures with explicit sample complexity. Conversely, no algorithm can circumvent the need for a number of proxy evaluations proportional to the inverse square of the desired utility tolerance if it aims to decide improvements reliably. We view this not as a pessimistic obstacle but as a guide for principled budgeting and for separating the roles of cost estimation and utility estimation.

Our contributions are accordingly threefold. First, we formalize iso-cost neighborhoods for micro-search on a given device and show how they remove the confounding present in unconstrained comparisons. Second, we present a layerwise search scheme that uses compensated micro-mutations to maintain iso-cost feasibility while adaptively allocating proxy evaluations to identify, with high probability, an approximately best choice within each local neighborhood. Third, we provide matching upper and lower bounds (up to constants and mild logarithmic factors) on the number of proxy evaluations required for such reliable local decisions under standard noise assumptions, together with an explicit accounting of how cost-model error inflates the iso-cost tolerance required for true feasibility.

The resulting perspective is intentionally modest in its optimization ambition: global optimality under hard cost budgets is intractable in general, and a local theory is the appropriate level at which one can make non-vacuous, device-specific claims. The practical implication is that micro-search can be made scientifically interpretable: when we claim an improvement, we can certify that it occurs at essentially fixed deployment cost on the target hardware, rather than as a byproduct of untracked budget changes.

2 Related work

Hardware-aware neural architecture search (NAS) has developed along three largely independent lines: (i) incorporating device-dependent cost signals during search, (ii) learning cost predictors that replace direct profiling, and (iii) addressing the statistical and optimization artifacts introduced by proxy evaluations and weight sharing. Our formulation of iso-cost micro-search is most directly connected to all three, but differs in the unit of comparison (iso-cost neighborhoods around a baseline) and in the type of guarantee sought (approximate *local* optimality at fixed hardware cost).

Early hardware-aware NAS methods introduced explicit latency or energy terms into the objective, either via scalarization (e.g., accuracy minus a weighted cost penalty) or via constraints (e.g., maximize accuracy subject to a latency budget) **???**. These approaches typically treat cost as a differentiable surrogate, a look-up-table estimate, or a black-box measurement invoked during search. Subsequent works emphasized the strong dependence of cost on the compilation stack and kernel selection, leading to device-specific search spaces and per-device adaptation **??**. Our work is aligned with the device-specific viewpoint: the cost function is indexed by the hardware target, and micro-choices are evaluated with respect to that target. However, rather than exploring a global constrained space, we focus on controlled local comparisons that hold cost fixed up to tolerance.

A central practical difficulty is the construction of accurate and query-efficient cost predictors. Many systems approximate latency by summing per-operator measurements from a look-up table **?**; others train regressors on architectural features and a profiling dataset **??**. Differentiable NAS variants sometimes incorporate a continuous relaxation of latency to enable gradient-based updates **??**. These estimators are known to exhibit systematic error when kernel fusion, tiling, and memory traffic dominate, and the error can be architecture-dependent. We do not attempt to eliminate this error; instead we treat the cost model as an estimator with explicit relative error and propagate that error into a slightly inflated iso-cost tolerance, which is the appropriate notion for feasibility when decisions are made adaptively.

Multi-objective NAS and Pareto-frontier methods provide a complementary perspective in which one seeks a set of trade-off solutions rather than a single optimum **??**. In practice, the Pareto approach is often instantiated by repeated constrained searches at different budgets or by maintaining a population under dominance relations. Our contribution is not a new multi-objective algorithm, but a local primitive that can be used as a subroutine: by anchoring at multiple baseline costs one may obtain a set of locally Pareto-efficient architectures, while retaining interpretability of each micro-change as a cost-controlled intervention.

The interaction between proxy evaluation and architecture selection has been extensively studied. Weight-sharing approaches (e.g., one-shot NAS) reduce cost by training a supernet and inheriting weights for sub-architectures **???**. It is now well documented that supernet ranking can be poorly correlated with stand-alone training due to coupling between candidates, optimization bias, and unfair training exposure **??**. Even without weight sharing, short-training proxies and partial datasets introduce significant noise and can mis-rank candidates when true performance gaps are small. Our framework treats proxy measurements as noisy observations and makes the selection step explicit as a fixed-confidence identification problem. This separates the algorithmic role of proxy training (utility estimation) from that of profiling or prediction (cost estimation), and it yields principled sample-complexity accounting.

Finally, our setup is related to macro–micro NAS decompositions in which a coarse macro-skeleton (stages, depths, resolutions) is fixed or searched at a higher level, while micro-choices (operators, kernel sizes, expansion ratios, attention windows) are refined locally **???**. We adopt this decomposition but concentrate on the micro stage under a fixed macro template, because this is the regime where confounding by cost inflation is most acute: micro-changes are often accepted based on small proxy gains that may be attributable to implicit budget shifts rather than genuine operator improvements. The iso-cost neighborhood formalism can be viewed as a device-specific control mechanism for the micro stage, independent of the particular macro-search strategy used upstream.

This discussion motivates the notation and problem statement in the next section, where we formalize the architecture family, the hardware-indexed cost function, the proxy evaluation model, and the notion of approximate local optimality under iso-cost neighborhoods.

3 Preliminaries and notation

We work in a *macro-fixed* micro-search setting. A baseline template specifies the stage layout (depth, resolutions, and connectivity) and identifies L mutable layers (or blocks) whose *micro-choices* may be altered. The resulting architecture family is a finite set $\mathcal{X}$: each $x \in \mathcal{X}$ is obtained by selecting, for every mutable layer $\ell \in \{1, \ldots, L\}$, an operator or kernel variant (e.g., depthwise separable versus standard convolution, kernel size, expansion ratio, attention window) together with admissible implementation-level knobs such as width parameters (channels) and numeric precision. We write $x = (x_1, \ldots, x_L)$ when emphasizing the layerwise decomposition, and we use $x_{-\ell}$ to denote all choices except those at layer ℓ.

A *hardware target* is denoted by h and represents a concrete device together with its runtime stack (compiler, libraries, and kernel implementations). Deployment cost is indexed by h: we let

$$g_h : \mathcal{X} \to \mathbb{R}_+$$

be the true cost of running x on h, where "cost" may refer to latency, energy, or activation memory, depending on the application. Throughout, we assume that g_h is (i) stagewise separable up to standard measurement noise, and (ii) monotone in the compensation knobs that we later use to enforce iso-cost comparisons; concretely, increasing width or using higher precision does not decrease cost. These assumptions are satisfied approximately for common inference pipelines and are the minimal structure needed for a well-posed compensation step.

Direct measurement of $g_h(x)$ on device is often expensive during search. We therefore assume access to a cost estimator (profiler, regressor, or lookup-table model)

$$\widehat{g}_h : \mathcal{X} \to \mathbb{R}_+,$$

which may be queried cheaply. Our analysis treats $\widehat{g}_h$ as accurate up to relative error: for each queried x, with probability at least $1 - \delta_g$ we have

$$\left|\widehat{g}_h(x) - g_h(x)\right| \le \eta\, g_h(x),$$

for some $\eta \in (0, 1)$ and per-query failure probability δ_g. We emphasize that we do not require unbiasedness or independence across x; the only property we use is a high-probability multiplicative bound on queried points.

Utility is measured by a true function $f : \mathcal{X} \to \mathbb{R}$ (e.g., accuracy, or accuracy minus a regularizer) under a fixed evaluation protocol. During micro-search we do not observe $f(x)$ directly; instead we obtain a noisy proxy $\widehat{f}(x)$ computed by a short-training routine (limited epochs, partial data, or otherwise truncated optimization). We model this as

$$\widehat{f}(x) = f(x) + \xi,$$

where ξ is mean-zero and σ-sub-Gaussian. In particular, for each $t \in \mathbb{R}$ we have $\mathbb{E}[\exp(t\xi)] \le \exp(\frac{1}{2}\sigma^2 t^2)$, and repeated evaluations of the same x yield independent copies of ξ. This abstraction covers the standard concentration tools used in fixed-confidence selection and will allow us to account explicitly for the proxy evaluation budget.

The primitive operation in our micro-search is a *proposal* π, which denotes a local edit at a single layer (e.g., "replace 3×3 conv by 5×5 conv at layer ℓ"). Applying π to x yields a tentative architecture x', which typically changes cost. To control for such changes we will enforce *iso-cost* comparisons relative to a baseline cost. Fix a tolerance $\tau \in (0, 1)$ and define the iso-cost band around x by

$$\mathcal{B}_{h,\tau}(x) \;=\; \Big\{x'' \in \mathcal{X} :\; g_h(x'') \in \big[(1-\tau)g_h(x), (1+\tau)g_h(x)\big]\Big\}.$$

In practice we implement this band using $\widehat{g}_h$ together with compensation knobs. We denote by w the (possibly vector-valued) width parameters that may be scaled within prescribed bounds, and by p the precision choice drawn from a finite set (e.g., $\{\mathrm{INT8}, \mathrm{FP16}\}$). When a proposal π increases estimated cost, we compensate by reducing width and/or precision (and conversely) to bring the modified architecture back into the band. We denote the resulting compensated operator by $M_{h,\tau}(x;\pi)$.

The induced iso-cost neighborhood of x is

$$\mathcal{N}_{h,\tau}(x) \;=\; \Big\{ M_{h,\tau}(x;\pi) : \; \pi \text{ allowed and compensation is feasible} \Big\},$$

where feasibility means that the compensated widths remain within $[w_{\min}, w_{\max}]$ and precisions remain in the allowed set. Finally, we formalize the local optimality notion we will target. Given $\varepsilon > 0$, we say that $\tilde{x}$ is an (ε, τ)-approximate local optimum (with respect to h and the allowed proposal set) if

$$f(\tilde{x}) \;\geq\; f(x'') - \varepsilon \qquad \text{for all } x'' \in \mathcal{N}_{h,\tau}(\tilde{x}).$$

Our algorithms will guarantee this property with probability at least $1-\delta$ for a user-specified failure probability δ, under a total proxy evaluation budget C (counted in training steps or epochs), which dominates computation in the micro-search stage.

4 Problem formulation

We study a *micro-search* problem in which the macro-structure is fixed and only local, layerwise choices are mutable. The central difficulty is that a micro-change proposal π typically alters deployment cost on the target hardware h, and hence naïve utility comparisons confound architectural merit with capacity or compute inflation. Our formulation therefore constrains all comparisons to be *iso-cost* (up to tolerance) by introducing an explicit compensation step.

(i) Iso-cost mutation feasibility. Fix a baseline architecture $x \in \mathcal{X}$ and a proposal π affecting a single layer ℓ. Let $\mathrm{Apply}(x;\pi)$ denote the tentative architecture obtained by applying π without compensation. We allow compensation through a prescribed set of knobs, consisting of (a subset of) width parameters w and possibly the precision choice p. We abstract a compensation choice by a variable c in a feasible set $\mathcal{C}$ encoding bounds (e.g. $w \in [w_{\min}, w_{\max}]$ and $p \in \mathcal{P}$). Let $x(\pi, c)$ denote the compensated architecture obtained from $\mathrm{Apply}(x;\pi)$ by applying c.

The *iso-cost feasibility* problem is:

$$\text{given } (x, \pi, h, \tau), \text{ find } c \in \mathcal{C} \text{ such that } \big|\widehat{g}_h(x(\pi, c)) - \widehat{g}_h(x)\big| \leq \tau\, \widehat{g}_h(x).$$

When such c exists we say that π is *(h, τ)-feasible at x.* The iso-cost mutation operator $M_{h,\tau}$ is any rule that returns a valid compensated architecture

$x' = M_{h,\tau}(x;\pi) = x(\pi, c)$ when feasible, and returns INFEASIBLE otherwise. In typical implementations, width compensation is (approximately) monotone in estimated cost, so c can be found by a one-dimensional search over a width scaling factor, optionally coupled with a small discrete scan over precision levels. We emphasize that feasibility is hardware-dependent: the same proposal may be feasible on one runtime stack and infeasible on another due to kernel availability or precision constraints.

(ii) Iso-cost local improvement under noisy utility. Given a baseline x, the *iso-cost neighborhood* $\mathcal{N}_{h,\tau}(x)$ is the set of all compensated candidates produced by applying allowed proposals and feasible compensation. Micro-search is then a constrained local optimization problem in which admissible moves are restricted to $\mathcal{N}_{h,\tau}(\cdot)$:

$$\max_{x \in \mathcal{X}} f(x) \quad \text{subject to taking only iso-cost moves on } h.$$

Since we only observe $\widehat{f}(x)$, we require a high-probability approximate optimality statement rather than exact ascent. Concretely, for tolerances (ε, τ) and failure probability δ, our target is to output an architecture $\tilde{x}$ satisfying

$$f(\tilde{x}) \;\geq\; f(x') - \varepsilon \qquad \forall\, x' \in \mathcal{N}_{h,\tau}(\tilde{x})$$

with probability at least $1 - \delta$, under a total proxy-evaluation budget C. This is a *local* guarantee: we do not seek a globally optimal architecture under a hard budget, but rather an architecture for which no single allowed compensated micro-change yields a utility gain exceeding ε. The role of the iso-cost constraint is operational as well as inferential: it ensures that comparisons between candidates may be attributed to the micro-choice itself, rather than to an unaccounted cost increase.

(iii) Iso-cost Pareto local search (optional extension). In some deployments, rather than fixing a single operating point, one seeks a small set of architectures that trade off utility and cost. Iso-cost neighborhoods can be used to construct an *approximately Pareto* set by repeating the local search around multiple *anchors*. Let $B_1 < \cdots < B_K$ be a collection of target costs (or baseline architectures $x_0^{(k)}$ with differing costs). For each anchor k, we run an iso-cost local search constrained to the band

$$g_h(x) \in [(1-\tau)B_k, (1+\tau)B_k],$$

returning $\tilde{x}^{(k)}$ that is an (ε, τ)-local optimum within that band. The resulting set $\{\tilde{x}^{(k)}\}_{k=1}^{K}$ yields a discrete approximation to the Pareto frontier in the sense that, near each anchor cost, $\tilde{x}^{(k)}$ cannot be locally improved without leaving the band. One may then post-process the collection to remove dominated points using $(f(\tilde{x}^{(k)}), \widehat{g}_h(\tilde{x}^{(k)}))$, while reserving occasional true measurements of g_h for final reporting. This extension preserves the same primitive operations—iso-cost mutation feasibility and noisy best-candidate selection—but changes the output from a single architecture to a set of locally optimal operating points across costs.

5 Cost modeling and calibration

Our micro-search procedure treats the deployment cost $g_h(x)$ on a fixed hardware target h as the relevant constraint variable. Concretely, $g_h(x)$ denotes the cost of running the full architecture x under a prescribed inference (or training) setting on h, where the metric may be latency, energy, or peak activation memory. We regard g_h as unknown to the algorithm, since it depends on kernel implementations, runtime fusion, precision support, and memory scheduling; moreover, direct evaluation on device is typically too expensive to perform for every candidate encountered during search.

Defining the measurement target. To avoid ambiguity, we fix a measurement protocol Prot_h that fully specifies: batch size, input resolution, sequence length (if applicable), number of warm-up iterations, number of timed iterations, clock/power settings, and whether measurements are end-to-end or exclude data transfer. We then define $g_h(x)$ as the (protocol-dependent) mean cost under repeated runs:

$$g_h(x) \;:=\; \mathbb{E}\big[\mathsf{Meas}_h(x; \mathsf{Prot}_h)\big],$$

where Meas_h is the raw observation (e.g. time per inference). In practice, the expectation is approximated by averaging multiple repetitions and optionally trimming outliers. For energy, Meas_h may combine duration with average power from a device-specific counter or external meter; for activation memory, Meas_h may be the maximum resident memory recorded by the runtime. While we present g_h as deterministic, the above definition makes explicit that measurement noise exists and is separated from modeling error.

Estimator structure. We assume access to an estimator $\widehat{g}_h(x)$ that is cheap to query. Typical constructions include (i) table-based summation of per-operator costs measured on h, optionally parameterized by tensor shapes and precision, and (ii) a learned regressor on architecture features with a small number of on-device calibration points. In both cases we aim to reflect the separability and monotonicity assumptions used later: stage-wise additivity (or bounded interaction) and non-decreasing cost in width and higher-precision choices. When the runtime admits graph-level fusion, additivity is only approximate; we therefore view separability as a modeling convenience and validate it empirically as part of calibration.

Relative-error requirement and its empirical meaning. Our analysis is phrased in terms of a multiplicative error bound: for architectures x that the algorithm queries,

$$\big|\widehat{g}_h(x) - g_h(x)\big| \;\le\; \eta\, g_h(x) \qquad \text{with probability at least } 1 - \delta_g.$$

This statement should be understood as an "auditability" condition: if we sample architectures according to the search distribution and occasionally measure their true cost on device, then the empirical relative error

$$e(x) \;:=\; \frac{|\widehat{g}_h(x) - g_h(x)|}{g_h(x)}$$

should be small for most queried x, and large deviations should occur with frequency at most δ_g (up to statistical uncertainty). The tolerance parameter τ used in iso-cost constraints is chosen *in addition* to η; later we will propagate the estimator error into a slightly inflated true-cost tolerance.

Calibration protocol. We recommend building a calibration set $\mathcal{Q} = \{x^{(1)}, \ldots, x^{(n)}\} \subset \mathcal{X}$ that covers the anticipated tensor-shape and precision regimes encountered in micro-search. For each $x^{(i)}$ we obtain a high-quality estimate of $g_h(x^{(i)})$ by repeated on-device measurement under Prot_h. We then fit (or adjust) $\widehat{g}_h$ on a training subset of $\mathcal{Q}$ and reserve an audit subset $\mathcal{Q}_{\text{test}}$ to estimate the tail of $e(x)$. A conservative choice is

$$\eta \;:=\; \text{Quantile}_{1-\delta_g}\big(\{e(x) : x \in \mathcal{Q}_{\text{test}}\}\big) \;+\; \gamma,$$

where $\gamma > 0$ is a safety margin accounting for finite-sample uncertainty and possible distribution shift between $\mathcal{Q}_{\text{test}}$ and architectures visited during search.

Testing monotonicity and diagnosing mismatch. Since our mutation operator will search over compensation knobs (e.g. width scaling or precision) using cost estimates, we empirically verify that $\widehat{g}_h$ is *approximately monotone* along these one-dimensional sweeps. For representative layers and widths $s_1 < s_2$, we check that $\widehat{g}_h(x(s_1)) \le \widehat{g}_h(x(s_2))$ and, more importantly, that the same inequality holds for measured g_h on a small audit subset. Violations typically indicate kernel regime switches (e.g. Winograd vs. direct convolution, tensor-core alignment thresholds, or different attention kernels), in which case we either refine the estimator around the threshold or restrict the allowable compensation range to remain within a stable regime.

Online calibration during search. Even with careful offline calibration, search may propose architectures outside the support of $\mathcal{Q}$. We therefore allow occasional true measurements of g_h on a small, adaptively selected set of visited candidates and update $\widehat{g}_h$ (or at least update the empirical estimate of η). This online step is not required by the algorithmic description, but it is the practical mechanism by which the stated η–δ_g condition is maintained as the queried set evolves. The subsequent section treats $\widehat{g}_h$ as fixed and accurate within (η, δ_g), and derives guarantees for iso-cost mutation under this calibration discipline.

6 Iso-cost micro-mutation operator

We now define the primitive that allows us to compare micro-architectural choices while controlling for deployment cost on a fixed target h. Fix a baseline architecture $x \in \mathcal{X}$ and a micro-change proposal π that acts at a designated layer (or block) ℓ; examples include changing the operator family (e.g. depthwise $\to$ regular convolution), changing a kernel hyperparameter (e.g. $3 \times 3 \to 5 \times 5$), or changing an attention window. Applying π to x yields a tentative architecture, which we denote by x^π, whose cost may differ substantially from that of x.

Compensation knobs and the iso-cost target. To enforce an iso-cost comparison, we allow a restricted set of compensation knobs, typically a width multiplier s applied to a designated subset of channels and/or a precision choice p from a finite set $\mathcal{P}$ (e.g. FP16/INT8). For notational convenience, we write $x^\pi(s, p)$ for the architecture obtained by applying proposal π and then applying compensation (s, p), with the convention that $s = 1$ and $p = p_0$ (the baseline precision) corresponds to no compensation. The goal is to find (s, p) such that

$$\left|\widehat{g}_h(x^\pi(s,p)) - \widehat{g}_h(x)\right| \;\le\; \tau\, \widehat{g}_h(x), \tag{1}$$

subject to feasibility constraints (bounds on s and admissible p), and then to return the compensated architecture

$$M_{h,\tau}(x;\pi) \;:=\; x^\pi(s^\star, p^\star).$$

This definition makes explicit that iso-cost is enforced with respect to the estimator $\widehat{g}_h$; the subsequent guarantee converts (1) into a statement about the true cost g_h.

Feasibility conditions. We require that the compensation search space be constrained in advance. For width scaling we assume an interval $s \in [s_{\min}, s_{\max}]$ such that the resulting tensor shapes remain valid and lie in regimes where the runtime is stable; in particular, we forbid scalings that change divisibility constraints required by vectorization or tensor-core alignment. For precision we assume a finite set $\mathcal{P}$ supported by the deployment stack on h and compatible with the operator at layer ℓ (e.g. some layers may not admit INT8 without additional quantization parameters). A proposal π is declared *infeasible* if no admissible compensation achieves (1).

Construction by one-dimensional search. Under the monotonicity assumption, the compensation search can be reduced to a small number of one-dimensional problems. The most common case is width-only compensation with fixed precision, where we assume that for the family $x^{\pi}(s)$ the true cost $g_h(x^{\pi}(s))$ is non-decreasing in s, and that $\widehat{g}_h(x^{\pi}(s))$ is sufficiently well-behaved to permit bracketing. We then search for $s^{\star}$ such that (1) holds, using either (i) binary search over a discretized grid of admissible widths, or (ii) a directed local search that steps s downwards if the estimated cost is too high and upwards otherwise. When both s and p are allowed, we first enumerate the small set of admissible precisions $p \in \mathcal{P}$, and for each p solve the width subproblem; we then select any feasible pair (s, p), breaking ties by maximizing a secondary criterion such as keeping s closest to 1 (to minimize representational drift away from the baseline).

Guarantee under cost-model error. Assume that on all queried architectures we have the multiplicative error condition

$$\left|\widehat{g}_h(z) - g_h(z)\right| \;\le\; \eta\, g_h(z),$$

with probability at least $1 - \delta_g$. Suppose that IsoCostMutate returns a compensated candidate $x' = x^{\pi}(s^{\star}, p^{\star})$ satisfying (1). Then, on the event that the above error bounds hold simultaneously for x and x', we can relate true costs via the inequalities

$$(1{-}\eta)g_h(x) \;\le\; \widehat{g}_h(x) \;\le\; (1{+}\eta)g_h(x), \qquad (1{-}\eta)g_h(x') \;\le\; \widehat{g}_h(x') \;\le\; (1{+}\eta)g_h(x').$$

Combining these with (1) yields a true iso-cost band of the form

$$g_h(x') \;\in\; \big[(1-\tau')g_h(x),\; (1+\tau')g_h(x)\big],$$

where τ' exceeds τ by an additive inflation on the order of η (e.g. one admissible explicit choice is $\tau' = \tau + 3\eta + 2\eta\tau$ for small η). Thus the operator $M_{h,\tau}$ enforces true iso-cost feasibility up to a predictable slack determined by calibration quality. In the next section we treat τ' as the effective tolerance and use it as an invariant when composing many such mutations inside a layerwise micro-search routine.

7 Layerwise iso-cost micro-search with confidence bounds

We now describe a concrete routine that composes the iso-cost mutation operator across layers while controlling proxy-evaluation noise. Fix a baseline

$x_0 \in \mathcal{X}$, tolerances $(\tau, \varepsilon, \delta)$, and a hardware target h. Our search proceeds layerwise: at each mutable layer $\ell \in \{1, \dots, L\}$ we enumerate a small set of admissible micro-change proposals $P_\ell = \{\pi_{\ell,1}, \dots, \pi_{\ell,m}\}$, construct the corresponding iso-cost compensated candidates, and then select among them using a fixed-confidence comparison rule under $\widehat{f}$.

Candidate set at a layer. Let x denote the incumbent architecture at the beginning of layer ℓ. For each proposal $\pi \in P_\ell$ we compute

$$x_\pi \;:=\; M_{h,\tau}(x; \pi),$$

discarding π if $M_{h,\tau}$ returns infeasible (i.e. no admissible compensation achieves the estimated iso-band). We then form the comparison set

$$S_\ell \;:=\; \{x\} \;\cup\; \{x_\pi :\; \pi \in P_\ell \text{ feasible}\}, \qquad |S_\ell| \leq m + 1.$$

All elements of S_ℓ are, by construction, comparable at (estimated) equal cost on h up to tolerance τ, and differ only in the local micro-choice plus compensating knob values.

Noisy selection by confidence bounds. We implement the selection step by repeated proxy evaluations. For each $z \in S_\ell$ we maintain an empirical mean $\overline{f}(z)$ from $n(z)$ i.i.d. proxy runs $\widehat{f}(z)$. Under the σ-sub-Gaussian assumption, a standard confidence radius is

$$r(n; \delta') \;:=\; \sigma \sqrt{\frac{2 \log(2/\delta')}{n}},$$

yielding (simultaneous) bounds $\overline{f}(z) \pm r(n(z); \delta')$ after $n(z)$ samples. A convenient fixed-confidence rule is to return any $\hat{z} \in S_\ell$ satisfying the termination condition

$$\mathrm{LCB}(\hat{z}) \;\geq\; \max_{z \in S_\ell \setminus \{\hat{z}\}} \mathrm{UCB}(z) \;-\; \varepsilon_\ell, \tag{2}$$

where $\mathrm{LCB}(z) = \overline{f}(z) - r(n(z); \delta_\ell/|S_\ell|)$ and $\mathrm{UCB}(z) = \overline{f}(z) + r(n(z); \delta_\ell/|S_\ell|)$. Condition (2) ensures that $\hat{z}$ is ε_ℓ-optimal within S_ℓ on the event that all confidence intervals hold. We may allocate samples uniformly (simplest) or adaptively by sampling the currently most ambiguous candidates (e.g. those with largest UCB among competitors).

Per-layer budgets and global parameters. To obtain an overall (ε, τ) guarantee after L layer updates, we choose schedules $(\varepsilon_\ell, \delta_\ell)$ with $\sum_{\ell=1}^{L} \varepsilon_\ell \leq \varepsilon$ and $\sum_{\ell=1}^{L} \delta_\ell \leq \delta$. A canonical choice is $\varepsilon_\ell = \varepsilon/L$ and $\delta_\ell = \delta/L$, which makes the union bound explicit and yields the sample complexity stated in §8. We additionally enforce an external compute budget C on the total number of proxy steps; if the selection subroutine would exceed the remaining budget, we terminate early and return the current incumbent (which weakens the formal ε guarantee but preserves the cost invariants).

Invariants maintained throughout the search. Writing x_ℓ for the incumbent after completing layer ℓ, the procedure maintains:

1. *Estimated iso-cost invariant:* by construction of $M_{h,\tau}$, each accepted update satisfies $|\widehat{g}_h(x_\ell) - \widehat{g}_h(x_{\ell-1})| \le \tau\, \widehat{g}_h(x_{\ell-1})$.
2. *True iso-cost invariant (high probability):* on the event that the cost-model error bounds hold for all queried architectures, the above implies a true band $g_h(x_\ell) \in [(1-\tau')g_h(x_{\ell-1}), (1+\tau')g_h(x_{\ell-1})]$ with $\tau' = \tau + O(\eta)$ as in §6.
3. *Feasibility invariant:* all widths/precisions remain within the admissible sets (divisibility, quantization support, and any layerwise constraints).
4. *Fairness invariant:* all candidates in each S_ℓ are evaluated using the same proxy protocol, so comparisons are attributable to micro-choices rather than evaluation artifacts.

Stopping criteria beyond fixed-confidence. In addition to budget exhaustion, we may stop early at a layer if the incumbent is already competitive, e.g. if

$$\mathrm{LCB}(x) \;\ge\; \max_{z \in S_\ell \setminus \{x\}} \mathrm{UCB}(z) \;-\; \varepsilon_\ell,$$

in which case no feasible iso-cost proposal at layer ℓ appears to improve utility by more than ε_ℓ given the accumulated evidence.

Optional multi-device variant. If we have a finite set of targets $\mathcal{H}$, we can enforce iso-cost simultaneously by defining a multi-constraint neighborhood

$$\mathcal{N}_{\mathcal{H},\tau}(x) \;:=\; \big\{x' : \forall h \in \mathcal{H},\; g_h(x') \in [(1-\tau)g_h(x), (1+\tau)g_h(x)]\big\},$$

and modifying $M_{h,\tau}$ to search for a common compensation (e.g. a shared width scaling and a per-device admissible precision, or a single precision supported by all $h \in \mathcal{H}$). In practice we implement this by checking the estimated band (1) for each $h \in \mathcal{H}$ and declaring infeasible unless all constraints pass; the subsequent analysis proceeds by replacing δ_g with a union bound over devices and treating the effective tolerance as the maximum inflation across $h \in \mathcal{H}$.

Fixed-confidence upper bound. We formalize the proxy-evaluation cost needed to obtain an (ε, τ)-approximate local optimum under the iso-cost neighborhood. Fix a layer ℓ and its comparison set S_ℓ of size $K_\ell := |S_\ell| \le m+1$. On the event that all confidence intervals $\overline{f}(z) \pm r(n(z); \delta_\ell/K_\ell)$ hold simultaneously for all $z \in S_\ell$ and all sampling times, any termination rule of the form (2) returns an ε_ℓ-optimal element of S_ℓ, i.e.,

$$f(\hat{z}_\ell) \;\ge\; \max_{z \in S_\ell} f(z) \;-\; \varepsilon_\ell.$$

A standard argument (sub-Gaussian concentration plus a union bound over arms and sampling rounds) implies that we can ensure this event with probability at least $1 - \delta_\ell$ by choosing radii $r(\cdot; \delta_\ell / K_\ell)$ and by sampling each candidate sufficiently many times. In particular, a uniform allocation $n(z) \equiv n_\ell$ with

$$n_\ell \;=\; O\!\left(\frac{\sigma^2}{\varepsilon_\ell^2} \log \frac{K_\ell}{\delta_\ell}\right)$$

suffices to guarantee that every interval half-width is at most $\varepsilon_\ell/2$, hence (2) holds for some $\hat{z}_\ell$ and yields ε_ℓ-optimality in S_ℓ. This yields a per-layer proxy-evaluation count of

$$O\!\left(K_\ell \cdot \frac{\sigma^2}{\varepsilon_\ell^2} \log \frac{K_\ell}{\delta_\ell}\right)$$

under the simplest sampling strategy; adaptive sampling (successive elimination or LUCB-style rules) can reduce constants and improve dependence on instance-specific gaps, but does not change the worst-case $\sigma^2 \varepsilon_\ell^{-2} \log(1/\delta_\ell)$ scaling.

From per-layer selection to local optimality. Let $x_{\ell-1}$ be the incumbent at the start of layer ℓ, and let x_ℓ be the selected architecture after running the fixed-confidence selector on S_ℓ. By construction, S_ℓ contains $x_{\ell-1}$ and all feasible one-step iso-cost mutations at layer ℓ (with compensation), so ε_ℓ-optimality in S_ℓ implies that no *feasible* proposal at layer ℓ improves upon x_ℓ by more than ε_ℓ when evaluated at the incumbent state. Summing this guarantee across layers gives, for any architecture x'' obtainable from the final $\tilde{x} := x_L$ by a single feasible iso-cost proposal at some layer, the bound

$$f(\tilde{x}) \;\ge\; f(x'') \;-\; \sum_{\ell=1}^{L} \varepsilon_\ell,$$

on the event that all per-layer selection guarantees hold. Choosing a schedule with $\sum_{\ell=1}^{L} \varepsilon_\ell \le \varepsilon$ and $\sum_{\ell=1}^{L} \delta_\ell \le \delta$ therefore yields the stated (ε, τ)-approximate local optimality with probability at least $1 - \delta$ (by a union bound over layers). The canonical choice $\varepsilon_\ell = \varepsilon/L$ and $\delta_\ell = \delta/L$ yields the total proxy-evaluation complexity

$$O\!\left(\sum_{\ell=1}^{L} K_\ell \cdot \frac{\sigma^2}{\varepsilon_\ell^2} \log \frac{K_\ell}{\delta_\ell}\right) \;=\; O\!\left(L \cdot (m+1) \cdot \frac{\sigma^2}{\varepsilon^2} \log \frac{L(m+1)}{\delta}\right),$$

up to universal constants, matching the dependence claimed earlier. We emphasize that this is a guarantee for a local optimum with respect to the iso-cost neighborhood induced by the allowed proposal set and compensation rule; it is not a statement about global optimality over $\mathcal{X}$.

Matching lower bound and tightness. The worst-case dependence on $(\sigma, \varepsilon, \delta)$ cannot be improved without additional structure. In the special case $L = 1$ and $|S_1| = 2$, with $g_h(x_0) = g_h(x_1)$ (exact iso-cost) and $\widehat{f}(x) = f(x)+\xi$ for σ-sub-Gaussian ξ, deciding which of x_0, x_1 is better by at least ε reduces to a two-hypothesis testing problem with mean gap ε. Standard information-theoretic reductions (Le Cam or a change-of-measure argument) show that any algorithm that returns an ε-optimal choice with probability at least $1-\delta$ must use

$$\Omega\left(\frac{\sigma^2}{\varepsilon^2}\log\frac{1}{\delta}\right)$$

proxy evaluations in expectation. Consequently, our upper bound is minimax-optimal in σ^2/ε^2 and in $\log(1/\delta)$, and the remaining multiplicative factor $L(m+1)$ is unavoidable in the worst case when each layer contributes an independent ambiguous choice.

Assumptions and their roles. The sub-Gaussian and independence assumptions are invoked only to obtain explicit confidence radii and a clean fixed-confidence stopping rule; other noise models (bounded, sub-exponential, or mildly dependent) can be accommodated by modifying radii and incurring corresponding constants. The cost-model accuracy assumption affects only feasibility of the iso-cost constraint: the selection analysis compares utilities within S_ℓ and is agnostic to g_h, while the translation from estimated iso-cost to true iso-cost inflates τ by a factor $O(\eta)$ as discussed earlier. Finally, we note that tighter bounds are possible in benign instances (large utility gaps, many dominated candidates), but without such gap conditions the lower bound shows that no algorithm can uniformly beat the stated scaling.

8 9. Practical evaluation plan (implementation-dependent): benchmarks, devices, cost metrics, baselines (parameter-normalization, cost-penalty search), and metrics (Pareto front quality, rank correlation, transfer).

We regard empirical validation as an implementation-dependent complement to the preceding guarantees, and we therefore separate (i) the choice of benchmark families and hardware targets, (ii) the measurement protocol for the "true" deployment costs g_h, and (iii) the comparative methodology against natural baselines that do not enforce iso-cost constraints. Concretely, we select a small number of macro-templates with L mutable layers (e.g. a mobile CNN stage template and a small vision transformer template) and instantiate micro-choice sets that reflect realistic operator alternatives (kernel size, expansion ratio, attention windowing, fused vs. unfused blocks, and optional precision changes). We run all methods on the same set of supervised tasks (e.g. ImageNet-1k for vision and a smaller proxy dataset for rapid iteration) with a fixed evaluation protocol for $\widehat{f}$ (identical optimizer, schedule, augmentation, and number of steps), so that differences in $\widehat{f}(x)$ are attributable to the architectural choice and not to training confounders.

For hardware targets, we treat h as the pair (device, runtime stack) and explicitly include heterogeneity that is known to affect micro-choices: a mobile CPU with a vendor BLAS / XNNPACK backend, a mobile GPU (e.g. OpenCL/Vulkan), and a server GPU with TensorRT/cuDNN; optionally an edge accelerator with compiler-based deployment. For each h we specify a primary cost metric (latency) and one secondary metric (energy or activation memory) to ensure that improvements are not achieved by shifting cost into an unmeasured resource. The true deployment cost $g_h(x)$ is obtained by compiling/exporting x into the target stack and benchmarking on-device using repeated runs with warm-up; we report median latency and an uncertainty estimate (e.g. median absolute deviation) to expose kernel-level variance. When energy is the target, we use an external power monitor or on-device counters when reliable; when activation memory is the target, we use runtime peak memory (or a conservative static upper bound) under a fixed batch size. This measurement protocol defines the reference against which $\widehat{g}_h$ is judged.

To validate the estimator assumptions used by the iso-cost mutation operator, we audit $\widehat{g}_h$ by measuring $(\widehat{g}_h(x), g_h(x))$ on a stratified sample of architectures spanning the width/precision bounds and a representative set of operator choices. We then report (a) relative error statistics to estimate a practical η and (b) rank correlation (Spearman/Kendall) between $\widehat{g}_h$ and g_h within an iso-cost band, since compensation relies on preserving order as width is adjusted. Because cost models are often biased in specific regions (e.g. depthwise kernels, small channel counts), we also include a lightweight calibration step (a small set of on-device measurements to fit a monotone correction) and report the before/after impact on feasibility, namely the empirical frequency with which a candidate satisfying $|\widehat{g}_h(x') - \widehat{g}_h(x)| \leq \tau\widehat{g}_h(x)$ also satisfies $g_h(x') \in [(1-\tau)g_h(x), (1+\tau)g_h(x)]$.

We compare against baselines that reflect common practices for controlling cost without explicit iso-cost neighborhoods. First, a parameter-normalization baseline: for each proposal π we adjust width to match parameter count (or FLOPs) rather than g_h, then evaluate with the same $\widehat{f}$ protocol; this tests whether hardware-specific iso-cost is necessary beyond crude capacity matching. Second, a cost-penalty (Lagrangian) baseline that optimizes $\widehat{f}(x) - \lambda \widehat{g}_h(x)$ over the same micro-choice space, with λ tuned to hit the target budget; this reflects penalty-based NAS and highlights sensitivity to multiplier selection. Third, a constrained search baseline that rejects candidates with $\widehat{g}_h(x) > B$ for a fixed budget B but allows arbitrary slack below B, which tests the confounding effect of "free" cost reductions. We additionally include random local mutation under the same evaluation budget C as a sanity check on sample efficiency.

We evaluate outcomes along three axes. (i) *Anchor-wise improvement*: for each initial anchor architecture x_0 we report the achieved utility $f(\tilde{x})$ (or a higher-fidelity surrogate) together with its measured cost $g_h(\tilde{x})$ and the iso-cost deviation relative to $g_h(x_0)$. (ii) *Pareto quality*: by repeating iso-cost search at multiple anchors (different budgets or different x_0 widths), we obtain a set of candidates and compute standard Pareto metrics (hypervolume and dominated fraction) under true costs g_h; this directly tests whether iso-cost local optima trace a meaningful frontier. (iii) *Predictive validity*: we report the correlation between proxy utility $\widehat{f}$ and final trained utility f (or a longer training protocol) to quantify proxy fidelity, and we measure the extent to which the ordering among iso-cost neighbors is preserved as training is extended.

Finally, we test transfer. Given a solution $\tilde{x}$ found on hardware h, we re-deploy the same micro-choices on a different hardware h' while re-running compensation (width/precision) to satisfy the iso-cost constraint on h', and we report the retained utility gain relative to the corresponding anchor on h'. This isolates whether the micro-choices are intrinsically beneficial or merely exploit idiosyncrasies of a particular kernel library. We also repeat the search across random seeds and across at least one dataset shift (e.g. ImageNet $\to$ a downstream classification task) to assess robustness of the local improvements.

Limitations and extensions. We record several respects in which the iso-cost formalism is not, by itself, a complete theory of deployment-aware micro-search, and we indicate extensions that preserve the central idea while relaxing assumptions that are convenient for analysis but imperfect in practice.

Non-monotone or irregular cost responses. Our use of compensation via a single width scaling parameter (and optional precision choice) tacitly appeals to a monotonic relationship $s \mapsto g_h(x(s))$, which justifies binary search and yields a clean propagation of the cost-model error in Thm. 1. Real systems may violate this monotonicity because of kernel selection thresholds, padding-induced layout changes, cache effects, or compiler heuristics; empirically, g_h can be *piecewise* monotone with discontinuities and small inversions. In such regimes, a direct extension is to replace binary search by a *finite compensation set* $\mathcal{S}$ (a small grid of admissible widths and/or a finite set of precisions) and define

$$M_{h,\tau}(x;\pi) \in \arg \min_{x'' \in \{x'(s,p):(s,p)\in\mathcal{S}\}} \left|\widehat{g}_h(x'') - \widehat{g}_h(x)\right| \quad \text{s.t. feasibility},$$

followed by rejection if the best achievable deviation exceeds $\tau\widehat{g}_h(x)$. This modification trades analytic simplicity for robustness: we no longer require global monotonicity, only that $\mathcal{S}$ is rich enough to approximate the iso-cost band. One may further assume a bounded number of inversions (a "K-quasi-monotone" model), in which case a bracketed search over $\mathcal{S}$ recovers an $O(K \log|\mathcal{S}|)$ cost-query bound. When non-monotonicity is severe, it becomes natural to insert occasional *true* cost measurements for a small subset of candidates and treat $\widehat{g}_h$ as a proposal mechanism rather than a hard gate.

Kernel-level implementation variance and stochastic costs. Even when the compiled graph is fixed, measured latency and energy are random due to OS scheduling, DVFS, thermal throttling, and nondeterministic kernel dispatch. Our definition of $g_h(x)$ as a deterministic quantity should then be interpreted as a functional of the underlying distribution, e.g. $g_h(x) = \mathbb{E}[\mathrm{lat}(x)]$ or a high quantile. This suggests an extension in which both $\widehat{f}(x)$ and the measured $\widehat{g}_h(x)$ are noisy, with separate concentration parameters. The iso-cost constraint can be enforced via confidence intervals: we accept x'' only if

$$\mathrm{LCB}\big(g_h(x'')\big) \geq (1-\tau)\,\mathrm{UCB}\big(g_h(x)\big) \quad \text{and} \quad \mathrm{UCB}\big(g_h(x'')\big) \leq (1+\tau)\,\mathrm{LCB}\big(g_h(x)\big),$$

where bounds are computed either from repeated on-device measurements or from a calibrated predictor with an uncertainty model. This change makes the feasibility statement probabilistic in a second dimension and forces an explicit allocation of measurement budget between utility and cost; however, it also prevents silent violations caused by rare but systematic kernel slowdowns.

Multi-objective iso-cost neighborhoods. Many deployments constrain more than one resource (e.g. latency and peak activation memory), in which case a scalar g_h is insufficient. A direct generalization is to take a vector cost $c_h(x) \in \mathbb{R}^k_+$ and define an iso-cost band componentwise:

$$\mathcal{N}_{h,\boldsymbol{\tau}}(x) = \Big\{x'' : \forall j \in \{1,\ldots,k\},\ c_{h,j}(x'') \in \big[(1-\tau_j)c_{h,j}(x), (1+\tau_j)c_{h,j}(x)\big]\Big\}.$$

All preceding constructions extend syntactically with $\widehat{g}_h$ replaced by $\widehat{c}_h$, although feasibility becomes stricter and compensation may require multiple knobs (e.g. width for compute, precision for memory). One may also define iso-cost sets via a norm constraint $\|\log c_h(x'') - \log c_h(x)\|_\infty \leq \tau$, which is equivalent to a multiplicative band and is convenient for error propagation. Finally, if one wishes to *optimize* multiple utilities (e.g. accuracy and robustness) under iso-cost, the appropriate outcome is a local Pareto set within $\mathcal{N}_{h,\boldsymbol{\tau}}(x)$; layerwise best-arm identification can be replaced by elimination with dominance checks, but the sample complexity necessarily scales with the number of nondominated candidates.

Coupling with dynamic proxy-training budgets. Our analysis fixes a proxy protocol and counts the number of independent evaluations, thereby treating each sample of $\widehat{f}(x)$ as having equal cost. In practice, we may trade training steps for variance reduction: letting $\widehat{f}_t(x)$ denote the proxy after t steps, we often have $\mathrm{Var}(\widehat{f}_t(x))$ decreasing in t while the bias (relative to final f) may increase or decrease depending on optimization dynamics. An extension is to combine iso-cost neighborhoods with multi-fidelity racing: within each layer, we begin with small t for all candidates, eliminate clearly suboptimal ones using confidence bounds, and allocate larger t only to survivors. The fairness invariant then becomes: candidates are compared at equal *current* fidelity before elimination, while the overall budget C is allocated adaptively. Formally, one may view this as best-arm identification with variable sampling costs and time-varying noise $\sigma(t)$, yielding improved constants (and sometimes improved effective sample complexity) without changing the core (ε, τ) notion.

These extensions do not alter the principal interpretation: iso-cost constraints remove a dominant confounder, but they must be instantiated with care when the cost surface is irregular, noisy, multi-dimensional, or when the proxy evaluator admits adaptive fidelity.

Conclusion: iso-cost as a fairness primitive for micro-search. We have treated deployment-aware micro-search as a problem in which a designer wishes to ascribe observed utility changes to *architectural* micro-decisions rather than to accidental changes in effective capacity induced by higher latency, energy, or memory. The central device is the iso-cost neighborhood $\mathcal{N}_{h,\tau}(x)$, together with a mutation operator that compensates each proposal by adjusting permitted knobs so that cost remains within a multiplicative band around the incumbent. In this view, the iso-cost constraint is not an afterthought appended to search; it is the mechanism that makes local comparisons meaningful on a fixed hardware target h.

The key conceptual consequence is that iso-cost acts as a *fairness primitive* for micro-search: when we compare two candidates inside the same band, the nuisance degree of freedom corresponding to spending more deployment budget is (approximately) removed, and the remaining differences can be attributed to the micro-choice itself. This fairness interpretation is operational rather than philosophical. It says that, under a fixed evaluation protocol for $\widehat{f}$, the algorithm's preference relation is constrained to be invariant to cost-inflation tricks, and thus a selected improvement is, by construction, not merely a larger model in disguise. The identifiability claim embodied by this restriction underlies the usefulness of iso-cost neighborhoods as a unit of scientific comparison: the same primitive supports both automated search and human-guided ablations.

On the algorithmic side, the layerwise scheme we analyzed can be read as a template: at each layer we generate at most m feasible iso-cost mutations and solve a noisy best-arm identification problem over the incumbent and these candidates. The guarantee is local—we do not claim global optimality in $\mathcal{X}$—but it is the correct granularity for micro-search under realistic budgets. In particular, an (ε, τ)-approximate local optimum is a statement that the algorithm has paid enough samples to rule out, with high probability, any *single* allowed compensated micro-change that would improve f by more than ε without leaving the iso-cost band. The matching lower bound in the simplest case emphasizes that this sample cost is not an artifact of the proof: even when the cost constraint is exact, reliable choice among near-tied candidates requires $\Omega(\sigma^2\varepsilon^{-2}\log(1/\delta))$ proxy evaluations.

For deployment-aware neural architecture search (NAS), the immediate implication is that one may decouple concerns that are often conflated. Macro-search may propose stage layouts and overall budgets, while micro-search, constrained by iso-cost, can refine operator choices, kernel variants, attention windows, or precision patterns without drifting away from the intended deployment point. Moreover, because the constraint is anchored to a particular h, the resulting architecture is explicitly *hardware-conditional*: the same macro template can induce distinct micro-optima on different devices, and the iso-cost formalism tells us how to compare those optima fairly within each device's own cost scale.

The same primitive also clarifies how to approximate cost–utility tradeoffs without turning micro-search into a fully multi-objective problem. By repeating iso-cost local optimization from several anchors (distinct baseline costs), one may assemble a discrete approximation to a Pareto frontier, where each point is locally stable under iso-cost mutations. This is often the right outcome for deployment: practitioners rarely need a single "best" network in the abstract, but rather a small menu of architectures that are each near-optimal around a prescribed operating regime and can be selected by product constraints.

Finally, we emphasize what our formalization is and is not. It is a statement about *comparisons* under controlled cost, and thus about the reliability of micro-search decisions under noisy proxy evaluation. It is not, by itself, a complete account of how to build accurate cost predictors, how to choose mutation sets that span the relevant design degrees of freedom, or how to guarantee global optimality under hard budgets. Nevertheless, as a primitive, iso-cost neighborhoods provide a principled interface between hardware models, search heuristics, and statistical decision rules. In our judgment, this interface is the right abstraction for deployment-aware NAS: it keeps the search honest with respect to the resource that ultimately matters, while remaining compatible with the practical realities of noisy evaluation and limited compute.