ApproxRoBoT: Training-Free NAS in Implicit Spaces via Quantile Candidate Generators

Liz Lemma Future Detective

January 20, 2026

Abstract

Training-free neural architecture search (NAS) replaces expensive training with zero-cost proxies, but existing methods—including RoBoT (ICLR 2024)—implicitly assume a tabular regime: they can compute proxy scores for all architectures and select arg max exactly. In modern 2026 settings (adapter composition for LLMs, routing policies, diffusion U-Nets), the search space is implicit and streaming: architectures are produced by generators, and exact maximization is unavailable. We formalize this setting through a quantile candidate generator $G(\theta)$ that samples architectures concentrated in the proxy top-L region, and we introduce ApproxRoBoT-GEN, which learns a robust proxy combiner and exploits it using only sampled candidates. Our main theoretical contribution is an end-to-end expected true-rank guarantee that (i) replaces exact $\arg\max$ with a generator quality parameter (L, δ) and (ii) quantifies a clean trade-off between generator effort and expensive objective evaluations by setting $L = \eta K$. We provide matching lower bounds showing the η dependence is unavoidable. Experiments (to be added) would validate the theory on large DARTS-like pools and on a modern implicit search task (e.g., LoRA/routing design), demonstrating that RoBoT-style robustness and boosting persist without enumeration.

Table of Contents

guarantee (L, δ) and rank-threshold observability; define evaluation budget and cost model (objective vs proxy vs generator compute).

4. 4. ApproxRoBoT-GEN algorithm: exploration (learn θ) + exploitation (evaluate K samples); engineering variants (batched sampling, duplicate handling, diversity) flagged as empirical add-ons.

5. 5. Main guarantees I (exploitation): expected true-rank bounds for best-of-K sampled candidates under (L, δ)-quantile generators; explicit dependence on ρ_L and δ; choosing $L = \eta K$ yields graceful degradation by η.

6. 6. Main guarantees II (learning θ under stochastic actions): bandit/BO guarantees for maximizing an implicit success probability $g_L(\theta)$ (or related surrogate) from noisy observations; plug-in to exploitation bound for an end-to-end theorem.

7. 7. Lower bounds and tightness: construct worst-case instances showing $\Omega(\eta)$ degradation is necessary; discuss when better-than-η is possible (additional generator assumptions).

8. 8. Experimental plan (implementation strengthens contribution): large finite pools (DARTS-like) with 'no-enumeration' protocol; implicit adapter/routing space; ablations over generator quality η, proxy set size, and exploration/exploitation splits.

9. 9. Discussion and limitations: realism of quantile generator assumption; relaxing conditional-uniformity; multi-objective constraints; nonstationary objectives; implications for proxy design.

10. 10. Conclusion: training-free NAS with guarantees in implicit spaces; open problems.

1 Introduction

Training-free neural architecture search (NAS) was originally developed under a *tabular* abstraction: one first specifies a finite candidate set $\mathcal{A}_{\text{tab}} \subset \mathcal{A}$, computes a collection of inexpensive proxy metrics $M_1, \dots, M_M$ for every $A \in \mathcal{A}_{\text{tab}}$, combines these metrics via a score $S_\theta(A) = \sum_{i=1}^{M} \theta_i M_i(A)$, and then returns architectures with large S_θ, optionally followed by a small number of expensive evaluations of the true objective $f(A)$. Within this abstraction, the operation $A(\theta) = \arg\max_{A \in \mathcal{A}_{\text{tab}}} S_\theta(A)$ is well-defined and computationally trivial given the table, and much of the algorithmic and empirical work reduces to learning a good θ and choosing a suitable evaluation budget.

By 2026, this abstraction is increasingly misaligned with practice. The candidate space $\mathcal{A}$ is often specified implicitly and is effectively unbounded for the purposes of search: architectures may be parameterized by compositional programs, conditional generation rules, hardware-conditional templates, or other forms of structured design where the set of feasible architectures is defined by constraints rather than enumeration. Moreover, in many modern pipelines the cost of *materializing* a candidate (i.e., producing its full specification and checking feasibility) is itself nontrivial, and the number of potential candidates grows super-exponentially with depth, width, and operator vocabulary. Consequently, the central primitive of tabular training-free NAS—sorting all candidates by S_θ—is no longer available. Even if each $M_i(A)$ is cheap for a given A, the set $\{M_i(A) : A \in \mathcal{A}\}$ cannot be computed globally, and proxy maximization must be performed under *sampling access* to candidates.

This shift from tabular to implicit search spaces changes the mathematical problem. We assume access to (i) metric oracles $M_i(A)$ that can be computed for sampled architectures A, (ii) an expensive objective oracle $f(A)$, and crucially (iii) a candidate generator $G(\theta)$ which, for a chosen parameter θ, returns a random architecture $A \sim D_\theta$. The generator may itself be implemented by a learned model, a heuristic constructor, or a constrained sampler; regardless of implementation, it constitutes the only feasible mechanism for exploring $\mathcal{A}$. In this setting, θ no longer merely defines a scoring function S_θ; rather, θ is an *action* that controls the distribution D_θ from which candidates are drawn. As a result, the naive surrogate-based reduction "maximize S_θ and then evaluate f" is ill-posed: the maximizer $\arg\max_{A \in \mathcal{A}} S_\theta(A)$ need not be accessible, and, more importantly, the distribution D_θ may assign negligible mass to the true proxy maximizers even when they exist.

The failure mode can be stated in rank-theoretic terms. Fix L and define the proxy top set $\text{Top}_L(\theta) = \{A \in \mathcal{A} : R_{S_\theta}(A) \le L\}$. In a tabular regime, retrieving $\text{Top}_L(\theta)$ is computationally straightforward once all proxy scores are known, and exploitation reduces to evaluating a few elements from this set under f. In an implicit regime, one must instead rely on the generator to *hit* $\text{Top}_L(\theta)$ with nontrivial probability. Thus, the relevant property is not that

S_θ is well-correlated with f in expectation, but that the pair $(S_\theta, G(\theta))$ yields a distribution D_θ whose mass concentrates in a region where f is large. Two distinct sources of error appear: (a) *proxy misalignment*, in which $\mathrm{Top}_L(\theta)$ contains few truly good architectures, and (b) *generator approximation*, in which D_θ fails to place sufficient probability on $\mathrm{Top}_L(\theta)$ itself. Either error can dominate, and neither is captured by analyses that presuppose exact top-K retrieval.

These considerations motivate an explicit separation between (i) the *combinatorial* ranking induced by S_θ and (ii) the *sampling* behavior induced by D_θ. We therefore frame the goal as minimizing the expected true rank $\mathbb{E}[R_f(A_{\mathrm{out}})]$ subject to a budget T of expensive oracle calls. A rank-based objective is natural in the present context: it is invariant to monotone rescalings of f, it accommodates heterogeneous objectives (e.g., accuracy subject to latency constraints), and it isolates the fundamental difficulty of identifying extreme quantiles of f under limited queries. In particular, the quantity of interest is not the regression accuracy of proxy metrics, but the probability of sampling from the high-performing region of $\mathcal{A}$ and the order-statistical improvement obtained by repeated sampling.

The algorithmic question is then: how do we choose θ adaptively, using only T evaluations of f, when each chosen θ produces a *random* architecture? This is a stochastic decision problem over a continuous (or high-dimensional) action space, where the reward of an action θ is a latent success probability $g_L(\theta) = \Pr_{A\sim D_\theta}[R_f(A) \le L]$ for an appropriate L. From this perspective, the generator converts proxy tuning into a bandit/BO problem with structured noise: even if θ is fixed, different calls to $G(\theta)$ yield different candidates and hence different objective values. A principled method must therefore handle stochastic outcomes, must account for generator-induced approximation error, and must translate any learned improvement in $g_L(\theta)$ into an end-to-end guarantee on the best architecture returned after exploitation.

Our contribution is to provide such a formulation and to analyze a simple exploration–exploitation strategy under explicit, checkable assumptions on generator quality and proxy–objective agreement in the top region. The central quantitative phenomenon is an *inflation* tradeoff: when exploitation is limited to K expensive evaluations, it is often necessary to target a proxy region of size $L = \eta K$ with $\eta > 1$ in order to ensure that the generator can reliably sample from it. This inflation enlarges the search region and inherently worsens the best achievable true rank after K samples; indeed, in the worst case the expected-rank degradation scales linearly with η. Thus, rather than treating imperfect top-K retrieval as an implementation detail, we treat it as a first-class parameter in the model and in the guarantees.

Finally, we emphasize that the implicit-space setting does not merely introduce technical inconvenience; it forces a different notion of robustness. Any claim that a training-free proxy "selects good architectures" must specify (a) how candidates are obtained and (b) how proxy information interacts with sampling and with the limited objective budget. The remainder of this work develops the necessary background and then presents an algorithm and analysis that make these dependencies explicit, culminating in end-to-end expected-rank bounds and matching lower bounds under generator-only access.

2 Background and related work

The starting point for much of training-free NAS is the observation that, for a fixed architecture A, a variety of inexpensive statistics $M_1(A), \dots, M_M(A)$ can be computed without (or with negligible) training, and that these statistics often exhibit nontrivial rank correlation with a downstream objective $f(A)$ such as accuracy, accuracy–latency tradeoffs, or a constrained score. A common operational pattern is to form a linear (or otherwise parameterized) combiner $S_\theta(A) = \sum_{i=1}^{M} \theta_i M_i(A)$ and to tune θ so that large values of S_θ correspond to large values of f on a calibration set. The RoBoT family of methods (in the broad sense of "robust training-free proxy tuning") instantiates this pattern with an explicit exploration–exploitation loop: one adaptively proposes θ (often via Bayesian optimization over θ), uses the proxy score S_θ to select promising architectures, queries f on those architectures, and then updates the belief over θ until the budget is exhausted.

This template is conceptually appealing because it reduces an enormous architecture search problem to a low-dimensional continuous optimization over θ. However, the reduction is exact only under a strong retrieval assumption that is often left implicit. In the tabular regime, one fixes a finite candidate set $\mathcal{A}_{\text{tab}}$, computes all $M_i(A)$, and then evaluates $A(\theta) = \arg\max_{A \in \mathcal{A}_{\text{tab}}} S_\theta(A)$ (or the entire top-K list) by sorting. In this regime, Bayesian optimization over θ indeed corresponds to choosing a proxy weighting and then deterministically extracting the architecture(s) that maximize the resulting proxy. The stochasticity is only in the (possibly noisy) observations of f, not in the act of retrieving a proxy maximizer. Accordingly, many empirical pipelines report performance as a function of how well the learned proxy ranking matches the true ranking on $\mathcal{A}_{\text{tab}}$, while treating the act of selecting top-K under S_θ as algorithmically trivial.

Once $\mathcal{A}$ is specified implicitly rather than enumerated, this retrieval assumption becomes the dominant issue. Even if θ were known, the map $\theta \mapsto A(\theta)$ need not be computable, and in fact the relevant operational primitive is not "compute $\arg\max S_\theta$" but "sample a candidate which is *likely* to have high S_θ". It is therefore natural to split the classical RoBoT template into two distinct components: (i) the *proxy alignment* problem of choosing θ so that the proxy top region contains truly good architectures, and (ii) the *retrieval* problem of accessing that proxy top region without enumeration. The latter problem is typically addressed by a generator—a learned or heuristic conditional sampler—which attempts to concentrate its distribution D_θ on architectures with large proxy scores. From a methodological perspective, this moves the role of θ from "weights in a scoring function" to "an action controlling a sampling distribution," and it moves the role of Bayesian optimization from selecting a point in $\mathcal{A}$ to selecting a distribution over $\mathcal{A}$.

This separation clarifies why rank-based notions such as Precision@K (or its top-L variants) are central but insufficient. If we define $\mathrm{Top}_L(\theta)$ to be the top-L architectures under S_θ, then the set-overlap ratio $\rho_L(\theta) = |\mathrm{Top}_L(\theta) \cap \mathrm{Top}_L(f)|/L$ measures how much of the proxy top region is truly good. In a tabular setting, high $\rho_L(\theta)$ nearly implies successful exploitation, because we can explicitly evaluate one or more members of $\mathrm{Top}_L(\theta)$. In an implicit setting, high $\rho_L(\theta)$ does not by itself imply anything unless the generator places nontrivial probability mass on $\mathrm{Top}_L(\theta)$; conversely, an excellent generator cannot compensate for a proxy whose top region is mostly misaligned. Thus we are forced to reason about *two* probabilities: the probability of landing in the proxy top set, and the conditional probability that such a landing also lies in the true top set. This is precisely the reason to elevate "top-region retrieval" to a first-class object of study, rather than treating it as an engineering detail.

The broader literature supplies each ingredient of this picture, but rarely analyzes them jointly. On the proxy side, a large body of work proposes zero-shot or training-free metrics, including gradient-based saliency measures, synaptic flow criteria, Jacobian or NTK-inspired scores, and graph/topology heuristics. A parallel line of work considers *ensembles* or *learned combinations* of such metrics, observing that different metrics are informative in different regimes, and that a small number of expensive evaluations can be used to calibrate the combination weights θ. On the generative side, "generative NAS" and constraint-aware architecture synthesis treat architectures as structured objects produced by a sampler (e.g., autoregressive models, programmatic generators, diffusion-style samplers, or grammar-based constructors) conditioned on desired attributes. These methods provide exactly the kind of oracle $G(\theta)$ that we require, but their guarantees are typically stated in distribution-matching terms (e.g., likelihood or constraint satisfaction) rather than in terms of quantile retrieval of proxy top sets. Finally, on the optimization side, Bayesian optimization and bandit algorithms in non-Euclidean or combinatorial domains develop surrogates over structured candidates and acquisition strategies under limited f-queries, but they often still presume a mechanism for optimizing the acquisition function over the candidate domain (which, in an implicit space, is again a retrieval problem).

Our goal is therefore not to introduce new proxy metrics, nor to advocate a particular generator architecture, but to formulate a minimal abstraction that captures the interaction between proxy alignment and generator retrieval under a strict objective-evaluation budget. We will treat the generator as an oracle inducing a distribution D_θ, and we will measure its adequacy by a quantile-type guarantee relative to $\mathrm{Top}_L(\theta)$. Likewise, we will treat Bayesian optimization over θ as a method for maximizing a latent success probability (the chance that a sample from D_θ lands in a high-performing region of f), rather than as a method for maximizing S_θ itself. This perspective directly motivates the formal model and assumptions introduced next: rank-based performance criteria, top-region overlap measures, and explicit budgeted interaction with $G(\theta)$ and f.

3 Problem setup and models

We work in an implicit architecture space $\mathcal{A}$, which may be finite but too large to enumerate, or genuinely infinite (e.g., countably infinite programs or graphs). An *architecture* is an element $A \in \mathcal{A}$. Our ultimate target is an expensive objective $f : \mathcal{A} \to \mathbb{R}$, where larger values are better. The expense of f models any procedure that requires substantial computation or data access (full training, heavy fine-tuning, costly simulation, or deployment-time measurement). We assume that f can only be accessed via queries at selected architectures, and that the total number of queries is budgeted.

To express performance in a way that is invariant to monotone transformations of f, we will use ranks. Let $R_f(A) \in \{1, 2, \dots\}$ denote the *true rank* of A under f over $\mathcal{A}$, with $R_f(A) = 1$ indicating global optimality. Formally, $R_f(A) = 1 + \big|\{A' \in \mathcal{A} : f(A') > f(A)\}\big|$, with an arbitrary but fixed tie-breaking rule when $f(A') = f(A)$. Likewise, for any scoring function $S : \mathcal{A} \to \mathbb{R}$, we write $R_S(A)$ for the rank induced by S. For an integer $L \geq 1$, define the top-L set under a score S by

$$\mathrm{Top}_L(S) \;=\; \{A \in \mathcal{A} : R_S(A) \leq L\}.$$

In particular, we abbreviate $\mathrm{Top}_L(f) = \mathrm{Top}_L(R_f)$, the set of truly best L architectures, and we will also write $\mathrm{Top}_L(\theta)$ for $\mathrm{Top}_L(S_\theta)$ when S_θ is the proxy score defined below.

The training-free information available to us is modeled by a collection of M cheap metric oracles $M_i : \mathcal{A} \to \mathbb{R}$ for $i \in \{1, \dots, M\}$. We think of $M_i(A)$ as a statistic computable for a given architecture A without invoking f (e.g., a structural measure, a gradient-based saliency score at initialization, or any other zero-shot proxy). We will combine these metrics linearly: for parameters $\theta = (\theta_1, \dots, \theta_M) \in \Theta \subseteq \mathbb{R}^M$,

$$S_\theta(A) \;=\; \sum_{i=1}^{M} \theta_i \, M_i(A).$$

The parameter set Θ can encode normalization or stability constraints (e.g., $\|\theta\|_1 \leq 1$, $\theta_i \in [-1, 1]$, or a simplex constraint). Our analysis will not depend on a particular choice beyond measurability and compactness assumptions customary in bandit/BO analyses.

The critical modeling feature is that, in an implicit $\mathcal{A}$, we cannot compute $\arg\max_{A \in \mathcal{A}} S_\theta(A)$ by exhaustive evaluation of S_θ. Instead, we assume access to a black-box *candidate generator* $G(\theta)$ which, given θ, outputs a random architecture $A \sim D_\theta$, where D_θ is an induced distribution over $\mathcal{A}$. We emphasize that $G(\theta)$ is not required to return the maximizer of S_θ; it only biases sampling toward architectures that score well under S_θ (or are intended to do so). The algorithmic interaction pattern is therefore: choose θ, sample $A \sim G(\theta)$, optionally compute $M_i(A)$ (and thus $S_\theta(A)$), and decide whether to pay for an f-evaluation at A. The ability to sample does not imply the ability to characterize D_θ explicitly; we treat $G(\theta)$ as an oracle.

To relate the generator to the proxy ranking, we adopt a quantile-type approximation guarantee. Fix $L \geq 1$ and $\delta \in [0, 1]$. We say that G is an (L, δ)-*quantile generator* (with respect to the proxy family $\{S_\theta\}_{\theta \in \Theta}$) if, for every $\theta \in \Theta$,

$$\Pr_{A \sim D_\theta}\big[A \in \mathrm{Top}_L(\theta)\big] \;\geq\; 1 - \delta.$$

Equivalently, with probability at least $1 - \delta$, a draw from D_θ lands in the proxy top-L region. We interpret δ as a *generator failure probability*: it captures approximation error, mode dropping, or any mismatch between the intended proxy conditioning and the generator's realized sampling distribution. The parameter L plays a dual role: it both sets the granularity of the proxy top region and, as will become explicit later, controls the intrinsic difficulty of extracting a truly top architecture from a proxy-concentrated distribution.

Proxy alignment with the true objective is captured by the overlap of top sets. For each θ and L, we define

$$\rho_L(\theta) \;=\; \frac{|\mathrm{Top}_L(\theta) \cap \mathrm{Top}_L(f)|}{L},$$

the Precision@L of the proxy ranking relative to the true ranking. While $\rho_L(\theta)$ is not directly observable (since $\mathrm{Top}_L(f)$ is unknown), it provides a convenient target quantity: a large $\rho_L(\theta)$ indicates that the proxy top region contains many truly strong architectures. The generator guarantee and $\rho_L(\theta)$ will jointly determine the probability that a sample $A \sim D_\theta$ is truly good.

Because our interaction with f is budgeted, we will measure complexity primarily by the number of objective queries. Let T be the total number of calls we may make to f. An algorithm is any adaptive procedure that, at round t, chooses θ_t as a measurable function of past observations, draws $A_t \sim D_{\theta_t}$, and optionally queries $f(A_t)$, subject to at most T such queries total. We allow the objective to be noisy: a query returns $y(A) = f(A) + \xi$, where ξ is mean-zero σ-sub-Gaussian noise, independent across queries conditioned on A. Noise is not essential for the definitions but will be convenient when we invoke bandit/BO tools.

In addition to observing $y(A)$, we will sometimes reason in terms of a rank-threshold label that abstracts "is this architecture among the top L under f?" Concretely, define $Z(A) \in \{0, 1\}$ with conditional expectation

$$\mathbb{E}[Z(A) \mid A] \;=\; \mathbf{1}\{R_f(A) \leq L\},$$

possibly perturbed by mean-zero σ-sub-Gaussian noise (e.g., a stochastic classifier of "top-L" status derived from noisy $y(A)$, or an indicator against an unknown but fixed threshold corresponding to the L-th order statistic of f). This *rank-threshold observability* model is a deliberate abstraction: it captures the fact that, for learning θ, we often only need coarse feedback about whether a sample is "good enough" rather than its exact f-value, and it allows us to view $g_L(\theta) = \Pr_{A\sim D_\theta}[R_f(A) \leq L]$ as a bandit reward function over θ.

Finally, although objective queries are the primary budgeted resource, in practice there are additional costs. We therefore distinguish: (i) objective cost, counted by the query budget T; (ii) proxy computation cost, typically $O(M)$ per sampled architecture to compute $M_i(A)$ and $S_\theta(A)$; and (iii) generator cost, the (possibly substantial) wall-clock expense of producing one sample from D_θ. Our theoretical statements will be parameterized by T (and by L, δ), while engineering variants may trade additional generator/proxy computation for better use of the limited objective budget (e.g., drawing many samples and only evaluating f on the most proxy-promising ones). Under this model, the goal is to output, after at most T objective evaluations, an architecture A_{out} with small expected true rank $\mathbb{E}[R_f(A_{\text{out}})]$, or equivalently high expected objective value.

4 ApproxRoBoT-GEN: exploration–exploitation with generator access

We now describe the procedure we analyze. The algorithm is organized around two observations. First, for any fixed proxy parameter θ, the only way to access high-scoring architectures in an implicit $\mathcal{A}$ is by sampling $A \sim D_\theta$ from the generator. Second, because objective queries are the dominant cost, it is natural to spend a small fraction of the objective budget to identify a *good* θ, and then spend the remaining budget sampling from $D_{\hat{\theta}}$ and taking the best objective value among the evaluated draws.

Inputs and budget split. Fix a total objective budget T. We choose an exploration budget $T_0 \in \{0, 1, \ldots, T\}$ and an exploitation budget $K = T - T_0$. We also fix a proxy top-set size L, which will later be chosen on the order of K (we will often write $L = \eta K$ for an inflation factor $\eta \geq 1$). The algorithm has oracle access to the generator $G(\theta)$ (hence to sampling from D_θ) and to the expensive evaluator f (possibly noisy), and it may optionally compute the proxy metrics $M_i(A)$ on sampled candidates.

Exploration: learning a good θ under stochastic outcomes. During exploration, at each round $t \in \{1, \ldots, T_0\}$ we choose a parameter $\theta_t \in \Theta$ as a function of past observations. We then sample an architecture $A_t \sim G(\theta_t)$ and query the objective to obtain y_t (or, in the rank-threshold abstraction, a label Z_t indicating whether A_t lies in the true top-L). The key point is that even for a fixed θ, the observed feedback is random due to the sampling $A \sim D_\theta$. Thus, exploration is naturally viewed as a stochastic bandit (or Bayesian optimization) problem over θ, where the latent reward of θ is the success probability $g_L(\theta) = \Pr_{A\sim D_\theta}[R_f(A) \leq L]$, and each play of θ returns one noisy Bernoulli-like observation via the sampled architecture.

Concretely, we may instantiate the exploration strategy with any optimistic or posterior-sampling rule that is standard in bandit/BO settings. For example, if we model $g_L(\theta)$ as a smooth function over Θ, we can maintain a surrogate model (e.g., a Gaussian process) updated on the pairs (θ_t, Z_t), and pick θ_t by an upper-confidence criterion. If instead we only observe objective values y_t, we can similarly use y_t directly (or a derived label $Z_t = \mathbf{1}\{y_t \geq \tau\}$ for a suitable threshold τ). Our analysis will only require that exploration returns some $\hat{\theta}$ whose $g_L(\hat{\theta})$ is close to the best value achievable in Θ, up to a learning error decaying with T_0.

At the end of exploration, we commit to a single parameter $\hat{\theta}$. A simple choice, and the one we will use for definiteness, is to take $\hat{\theta}$ as the parameter among $\{\theta_t\}_{t\leq T_0}$ whose sampled architecture attained the largest observed objective value. When exploration is formulated as a bandit on Z_t, one can equivalently select the empirically best θ_t in terms of observed labels, or the maximizer of a posterior mean; these variants change constants but not the qualitative structure of the subsequent guarantees.

Exploitation: best-of-K objective evaluations from $D_{\hat{\theta}}$. In exploitation, we fix $\hat{\theta}$ and perform K further objective queries. In the most basic form, for each $j \in \{1, \ldots, K\}$ we draw $A_j \sim G(\hat{\theta})$, evaluate $f(A_j)$, and return

$$A_{\text{out}} \in \arg\max_{1\leq j\leq K} f(A_j),$$

with ties broken arbitrarily. This stage is intentionally austere: it makes no attempt to further adapt θ, and it uses objective evaluations only to choose the best architecture among sampled candidates. The reason for this design is that, under the quantile-generator model, the distribution $D_{\hat{\theta}}$ is already concentrated on architectures with high proxy score, and the remaining uncertainty is primarily whether these proxy-high samples coincide with truly good architectures. Theoretical bounds in the next section will treat each draw as an independent chance of landing in the true top-L set and then analyze the order statistics of the induced ranks.

Engineering variants (empirical add-ons). Several practical modifications are often beneficial, and we separate them conceptually from the core procedure above because they rely on additional computational resources (generator calls and proxy computations) or on properties of the generator not encoded in our minimal model.

Batched preselection within exploitation. Rather than spending one objective query per generator draw, we may draw a batch of $B \gg 1$ candidates $A^{(1)}, \dots, A^{(B)} \sim D_{\hat{\theta}}$, compute their proxy scores $S_{\hat{\theta}}(A^{(b)})$ (or a related score), and only evaluate f on the top-1 candidate (or top-b_0 candidates) in that batch. Repeating this procedure K times uses the same objective budget but increases the proxy-computation and generator costs by a factor of B. While our main theorems will focus on the simpler best-of-K sampling for clarity, the batched variant can be interpreted as increasing the effective success probability of each objective evaluation by conditioning on an internal proxy-based selection step.

Duplicate handling. A generator may return the same architecture multiple times, which can waste objective queries if not handled. In implementations, we can maintain a cache of previously evaluated architectures and either (i) skip objective evaluation when a duplicate is observed, replacing it by a fresh draw, or (ii) treat duplicates as repeated noisy evaluations and average the resulting y-values. Our formal model counts only objective queries, so (i) effectively increases the number of *distinct* evaluated architectures at fixed budget, whereas (ii) reduces noise. Since neither behavior is guaranteed by the oracle model, we will treat duplicate handling as an implementation detail; when duplicates are frequent, they can be conservatively viewed as reducing the effective exploitation budget below K.

Diversity and mode coverage. Especially when $D_{\hat{\theta}}$ is highly concentrated, it can be useful to enforce diversity among sampled candidates, for example by penalizing similarity to previously seen architectures, by sampling from a mixture over nearby θ's, or by reranking batches using a diversity-aware criterion. Such heuristics aim to mitigate mode collapse in $G(\theta)$ and increase coverage of $\mathrm{Top}_L(\hat{\theta})$, but they introduce additional assumptions (a similarity metric on $\mathcal{A}$, or control over the generator) beyond our present abstraction.

Parallelism. Both exploration and exploitation can be executed in parallel by selecting multiple θ's (or multiple draws from $D_{\hat{\theta}}$) before observing their objective outcomes. This affects wall-clock time but not the accounting of objective queries, and the guarantees we state later are formulated in terms of T_0 and K regardless of scheduling.

In summary, ApproxRoBoT-GEN consists of (i) a stochastic search over proxy parameters θ driven by objective feedback on generator samples and (ii) a final best-of-K selection from $D_{\hat{\theta}}$, with optional proxy-heavy preselection and other heuristics that we view as orthogonal empirical enhancements.

5 Main guarantees I (exploitation): best-of-K ranks under quantile generators

We first isolate the exploitation stage, treating the proxy parameter θ as fixed, and analyze the true rank of the best architecture returned after K objective evaluations drawn from the generator distribution D_θ. The resulting bounds make explicit how performance depends on (i) the generator's ability to concentrate on the proxy top set (through δ), and (ii) the alignment between proxy and objective rankings (through $\rho_L(\theta)$).

From proxy concentration to true-top-L hit probability. Fix $L \in \mathbb{N}$ and $\theta \in \Theta$. The (L, δ)-quantile generator guarantee asserts that

$$\Pr_{A \sim D_\theta}\big[A \in \mathrm{Top}_L(\theta)\big] \;\geq\; 1 - \delta. \tag{1}$$

This statement alone is agnostic to the objective f; it only certifies that the generator returns proxy-high candidates with high probability. To convert (1) into a lower bound on the probability of sampling a *truly* good architecture, we introduce the proxy–objective overlap (precision@L)

$$\rho_L(\theta) \;=\; \frac{\big|\mathrm{Top}_L(\theta) \cap \mathrm{Top}_L(f)\big|}{L}.$$

A direct decomposition yields, for the success probability

$$g_L(\theta) \;=\; \Pr_{A \sim D_\theta}\big[R_f(A) \leq L\big] \;=\; \Pr[A \in \mathrm{Top}_L(\theta)] \cdot \Pr\big[R_f(A) \leq L \,\big|\, A \in \mathrm{Top}_L(\theta)\big].$$

The first factor is controlled by δ via (1). The second factor depends on how D_θ allocates mass *within* $\mathrm{Top}_L(\theta)$. In the benign case that, conditional on $A \in \mathrm{Top}_L(\theta)$, the generator is (approximately) uniform over $\mathrm{Top}_L(\theta)$, we obtain the convenient lower bound

$$g_L(\theta) \;\geq\; (1 - \delta)\, \rho_L(\theta). \tag{2}$$

More generally, even without conditional uniformity, one can replace $\rho_L(\theta)$ in (2) by the conditional mass that D_θ assigns to $\mathrm{Top}_L(\theta) \cap \mathrm{Top}_L(f)$ given $A \in \mathrm{Top}_L(\theta)$; our subsequent exploitation analysis only requires a lower bound $p \leq g_L(\theta)$.

Best-of-K exploitation as order statistics under random successes. Consider exploitation with fixed θ: draw $A_1, \dots, A_K \overset{\text{iid}}{\sim} D_\theta$, evaluate $f(A_i)$, and return $A_{\text{out}} \in \arg\max_{i \le K} f(A_i)$. Let $R_i = R_f(A_i)$ denote the true ranks. We analyze $\mathbb{E}[R_f(A_{\text{out}})] = \mathbb{E}[\min_i R_i]$ under a rank-threshold model that separates two effects: (a) how often we hit the true top-L, and (b) how good the hit is when it happens.

Assume that each draw is a "success" (lies in $\text{Top}_L(f)$) with probability at least p, i.e.,

$$\Pr[R_i \le L] \;\ge\; p \qquad \text{for each } i \in \{1, \dots, K\}, \tag{3}$$

and assume further that conditional on success, the rank is uniform over $\{1, \dots, L\}$:

$$(R_i \mid R_i \le L) \;\sim\; \text{Unif}\{1, \dots, L\}. \tag{4}$$

The uniformity (4) is a stylized but analytically transparent way to express that, within the true top-L, the generator does not further privilege particular ranks. Let $X = \sum_{i=1}^{K} \mathbf{1}\{R_i \le L\} \sim \text{Binomial}(K, p)$ be the number of successes. Conditional on $X = x$, the best returned rank is the minimum of x i.i.d. uniform ranks in $\{1, \dots, L\}$, whose expectation is $(L+1)/(x+1)$. Taking expectation over X yields the explicit identity

$$\mathbb{E}[R_f(A_{\text{out}})] \;=\; (L+1)\,\mathbb{E}\left[\frac{1}{X+1}\right] \;=\; (L+1)\,\frac{1-(1-p)^{K+1}}{(K+1)p}. \tag{5}$$

Two immediate corollaries are worth recording. First, the probability that exploitation returns *some* truly top-L architecture is

$$\Pr\big[R_f(A_{\text{out}}) \le L\big] \;=\; 1-(1-p)^K, \tag{6}$$

which exhibits the expected geometric improvement in K at fixed p. Second, when $Kp \ge 1$, (5) implies the simpler upper bound

$$\mathbb{E}[R_f(A_{\text{out}})] \;\le\; \frac{2\,(L+1)}{(K+1)p}, \tag{7}$$

so the expected rank decays essentially like $1/(Kp)$, up to the scale factor L.

Explicit dependence on $\rho_L(\theta)$ and δ. Combining (2) with (5) (or (7)) yields a bound in the natural parameters of the proxy–generator interface. Under conditional uniformity within $\text{Top}_L(\theta)$ and using $p = (1-\delta)\rho_L(\theta)$, we have

$$\mathbb{E}[R_f(A_{\text{out}})] \;\le\; (L+1)\,\frac{1-\big(1-(1-\delta)\rho_L(\theta)\big)^{K+1}}{(K+1)(1-\delta)\rho_L(\theta)}. \tag{8}$$

Thus, degradation is multiplicative in $(1-\delta)^{-1}$ and $\rho_L(\theta)^{-1}$: failures to enter the proxy top set and misalignment between proxy and objective both reduce the effective success probability p, and exploitation cannot distinguish these two failure modes.

Choosing $L = \eta K$: graceful η-degradation. We finally highlight the commonly used scaling $L = \eta K$ with inflation factor $\eta \geq 1$, which formalizes the idea that the generator can only approximate a proxy maximizer up to a top-L region whose size grows with the number of objective evaluations. Substituting $L = \eta K$ into (7) yields, for $Kp \geq 1$,

$$\mathbb{E}[R_f(A_{\text{out}})] \;\leq\; \frac{2(\eta K + 1)}{(K+1)p} \;=\; O\left(\frac{\eta}{p}\right), \tag{9}$$

up to lower-order terms in K. In particular, if p does not deteriorate as K grows (e.g., the proxy precision $\rho_L(\theta)$ remains bounded away from zero at the chosen L), then the expected true rank scales at most linearly in η. This is the sense in which approximation in the generator—modeled here by enlarging the proxy top set from K to ηK—induces a controlled and interpretable loss in the final true rank. The next section addresses how, during exploration, we can select $\hat{\theta}$ so as to make $p = g_L(\hat{\theta})$ as large as possible under stochastic generator outcomes.

6 Main guarantees II (exploration): learning θ under stochastic actions

We now study the exploration stage, in which we adaptively choose proxy-combiner parameters θ_t while only observing objective information through architectures sampled from the induced distribution D_{θ_t}. The central difficulty is that the "action" θ does not deterministically produce an architecture, but rather a random draw $A \sim D_\theta$. Consequently, even if we intend to optimize the success probability $g_L(\theta) = \Pr_{A\sim D_\theta}[R_f(A) \leq L]$, each query yields only a noisy, single-sample estimate of this quantity.

A rank-threshold observation model. To connect exploration directly to the exploitation analysis, we assume that an expensive query returns a stochastic label $Z(A) \in \{0,1\}$ indicating whether the sampled architecture is truly in the top-L set. Formally, when we sample $A_t \sim D_{\theta_t}$ and query the objective oracle, we observe

$$Z_t \;=\; Z(A_t) \;=\; \mathbf{1}\{R_f(A_t) \leq L\} + \varepsilon_t, \tag{10}$$

where ε_t is conditionally mean-zero and σ-sub-Gaussian given the past. Taking expectation over the generator randomness, we obtain the induced mean-reward function on parameters,

$$g_L(\theta) \;=\; \mathbb{E}_{A\sim D_\theta}[\mathbf{1}\{R_f(A) \leq L\}] \;=\; \Pr_{A\sim D_\theta}[R_f(A) \leq L], \tag{11}$$

so that $\mathbb{E}[Z_t \mid \theta_t] = g_L(\theta_t)$. In this way, exploration reduces to a stochastic bandit optimization problem over Θ, with reward function $g_L(\cdot)$ that is expensive to sample but whose maximizer corresponds to the parameter that most frequently produces truly good architectures.

Modeling g_L as a smooth black-box function of θ. To obtain nonasymptotic guarantees for optimizing $g_L(\theta)$ from noisy samples, we impose a standard regularity assumption from Bayesian optimization. Namely, we assume g_L lies in the reproducing kernel Hilbert space (RKHS) associated with a positive definite kernel k on Θ, with bounded RKHS norm. This permits the use of GP-based upper confidence bound algorithms (GP-UCB) to select θ_t adaptively. The stochasticity arising from sampling $A_t \sim D_{\theta_t}$ is absorbed into the bandit noise: for fixed θ, the random variable $\mathbf{1}\{R_f(A) \le L\}$ is Bernoulli with mean $g_L(\theta)$, hence sub-Gaussian, and additional oracle noise ε_t is also sub-Gaussian by assumption. Therefore the total observation noise remains sub-Gaussian, and the usual GP-UCB analysis applies.

Concretely, running GP-UCB for T_0 exploration rounds produces a sequence $\theta_1, \dots, \theta_{T_0}$ and observations $Z_1, \dots, Z_{T_0}$. Let $\hat{\theta}$ be the best explored parameter according to observed rewards (or equivalently the best posterior mean, up to standard discretization issues). Then, with probability at least $1 - \delta_0$, we obtain a best-arm guarantee of the form

$$g_L(\hat{\theta}) \;\ge\; \max_{\theta \in \Theta} g_L(\theta) \;-\; \tilde{O}\left(\sqrt{\frac{\gamma_{T_0}}{T_0}}\right), \tag{12}$$

where γ_{T_0} is the maximum information gain for kernel k after T_0 samples. This is the appropriate notion of "learning θ" in our implicit-search setting: we are not learning a surrogate f over $\mathcal{A}$, but rather learning which generator parameter induces a distribution that most often produces truly good candidates.

From optimizing g_L to proxy alignment. The exploitation bounds require a lower bound $p \le \Pr_{A \sim D_{\hat{\theta}}}[R_f(A) \le L] = g_L(\hat{\theta})$. Equation (12) supplies such a bound relative to $\max_\theta g_L(\theta)$. In many instances, however, it is more interpretable to express success probability in terms of proxy precision $\rho_L(\theta)$ and generator quality δ. Under the conditional-uniformity condition within $\mathrm{Top}_L(\theta)$ (as in the exploitation analysis), we have the comparison

$$g_L(\theta) \;\ge\; (1 - \delta)\rho_L(\theta), \tag{13}$$

so maximizing $g_L(\theta)$ indirectly promotes high precision $\rho_L(\theta)$. More generally, if conditional uniformity fails, one may replace $\rho_L(\theta)$ by the conditional mass that D_θ assigns to $\mathrm{Top}_L(\theta) \cap \mathrm{Top}_L(f)$ given $A \in \mathrm{Top}_L(\theta)$; the learning procedure remains unchanged, since it only ever observes the realized labels Z_t.

In order to convert the exploration guarantee (12) into a statement about $\rho_L(\hat{\theta})$, we may invoke a calibration condition on the explored region, asserting that $g_L(\theta)$ is (approximately) monotone in $\rho_L(\theta)$ or at least that near-optimizers of g_L are near-optimizers of ρ_L. This condition is empirically checkable: during exploration, one can estimate $\rho_L(\theta_t)$ on held-out samples using a small number of additional objective queries, or validate that increasing observed success frequency corresponds to improved proxy–objective overlap.

End-to-end plug-in to exploitation. Having learned $\hat{\theta}$, we run exploitation by sampling K candidates from $D_{\hat{\theta}}$ and returning the best under f. The exploitation analysis applies with success parameter $p_{\hat{\theta}} = g_L(\hat{\theta})$, yielding an explicit expression for $\mathbb{E}[R_f(A_{\text{out}})]$ in terms of $p_{\hat{\theta}}$ and (K, L). Substituting the exploration guarantee (12) yields an end-to-end bound in which the final expected rank depends on (i) the intrinsic difficulty of finding θ that maximizes g_L, quantified by γ_{T_0} and T_0, and (ii) the unavoidable sampling difficulty during exploitation, captured by the dependence on K and $p_{\hat{\theta}}$. In particular, taking $L = \eta K$ makes the dependence on generator approximation explicit through the inflation factor η, and separates the two roles of budget: T_0 controls the statistical error in learning $\hat{\theta}$, while K controls the best-of-K improvement once $\hat{\theta}$ is fixed.

The next section shows that, without assumptions beyond quantile-generation and the limited feedback model above, the linear η-dependence is not merely an artifact of our analysis: it is information-theoretically necessary in the worst case, and can only be improved by strengthening the generator model or by introducing additional structure linking proxy scores to true performance.

7 Lower bounds and tightness: why linear η is unavoidable

We now show that the η-inflation appearing in the exploitation bound is not an artifact of analysis but a genuine information-theoretic obstruction under our access model. Concretely, when we choose $L = \eta K$ and restrict ourselves to obtaining candidates only through an (L, δ)-quantile generator, there exist instances in which no algorithm can achieve expected true rank $o(\eta)$ after K expensive queries.

A worst-case instance reduces to unstructured search within a size-L set. Fix any $K \geq 1$ and $L \geq K$. We construct a finite subset $\mathcal{B} \subset \mathcal{A}$ with $|\mathcal{B}| = L$ and define both the proxy metrics and the generator family so that (i) the proxy top set $\mathrm{Top}_L(\theta)$ is always exactly $\mathcal{B}$, and (ii) the generator is "perfect" in the sense that it samples uniformly from $\mathcal{B}$ (hence $\delta = 0$).

In such a case, any algorithm—regardless of how it adaptively selects θ_t—only ever observes i.i.d. samples from the uniform distribution on $\mathcal{B}$ when it requests a candidate to evaluate.

Formally, we may define the training-free metrics to be completely uninformative, e.g. $M_i(A) \equiv 0$ for all $A \in \mathcal{A}$ and all $i \in [M]$. Then $S_\theta(A) \equiv 0$ for all (θ, A), so the proxy ranking is arbitrary; we may fix a tie-breaking rule that declares $\mathrm{Top}_L(\theta) = \mathcal{B}$ for all θ. Define D_θ to be the uniform distribution on $\mathcal{B}$ for all θ. This satisfies the quantile-generation property with $\delta = 0$, since

$$\Pr_{A \sim D_\theta}\big[A \in \mathrm{Top}_L(\theta)\big] \;=\; 1.$$

Finally, assign the true objective f (equivalently the true ranks R_f) so that $\mathcal{B}$ contains architectures of all ranks $1, 2, \ldots, L$ (and any remaining architectures in $\mathcal{A} \setminus \mathcal{B}$ are worse than rank L). Under this construction, the algorithm receives no side information from S_θ (or from θ) and can only learn by directly querying f on sampled elements of $\mathcal{B}$.

Expected best rank from K uniform samples is $\Theta(L/K)$. Let $A_1, \ldots, A_K$ be the architectures queried by the algorithm (possibly with adaptively chosen θ_t), and let

$$R_{\min} \;=\; \min_{1 \le t \le K} R_f(A_t)$$

be the true rank of the returned architecture A_{out} (since the algorithm can do no better than output the best it has seen). In our instance, regardless of the algorithm, the random variables $R_f(A_t)$ are i.i.d. uniform on $\{1, \ldots, L\}$. Hence $R_{\min}$ is the minimum of K i.i.d. discrete uniforms. A standard order-statistics calculation yields

$$\mathbb{E}[R_{\min}] \;=\; \frac{L+1}{K+1}. \tag{14}$$

One way to see (14) is via the tail-sum formula:

$$\mathbb{E}[R_{\min}] \;=\; \sum_{r=1}^{L} \Pr[R_{\min} \ge r] \;=\; \sum_{r=1}^{L} \left(\frac{L-r+1}{L}\right)^{K} \;=\; \frac{1}{L^K} \sum_{s=1}^{L} s^K,$$

and the identity $\sum_{s=1}^{L} s^K = \frac{L^{K+1}}{K+1} + \frac{L^K}{2} + O(L^{K-1})$ implies $\mathbb{E}[R_{\min}] = \frac{L}{K+1} + O(1)$, consistent with (14) (indeed, the discrete uniform case admits the exact expression shown).

Setting $L = \eta K$ gives

$$\mathbb{E}[R_f(A_{\text{out}})] \;\ge\; \mathbb{E}[R_{\min}] \;=\; \frac{\eta K + 1}{K+1} \;\ge\; c\eta \qquad \text{for all } K \ge 1,$$

for a universal constant $c > 0$. This proves that $\Omega(\eta)$ expected-rank degradation is unavoidable even with a perfect ($\delta = 0$) generator, provided the proxy carries no information that allows nonuniform sampling within the proxy top set.

Tightness relative to our upper bounds. The lower bound above matches the scaling in the exploitation guarantee when the success probability p is constant. Indeed, in the idealized setting where every sample lies in the true top-L region (so $p = 1$ in Theorem 2), the expected best rank from K draws is $\Theta(L/K)$, and with $L = \eta K$ this becomes $\Theta(\eta)$. Thus, absent additional structure, one should not expect a sublinear dependence on η from any method that ultimately relies on selecting the best among K objective-evaluated samples drawn from a proxy-defined top-L region.

When can one do better than linear η? Improving upon $\Theta(\eta)$ requires strengthening the model beyond quantile generation plus expensive objective access. We highlight several sufficient directions.

(i) Nonuniformity favoring high true rank within $\mathrm{Top}_L(\theta)$. If, conditional on landing in $\mathrm{Top}_L(\theta)$, the generator distribution places disproportionately large mass on $\mathrm{Top}_L(\theta) \cap \mathrm{Top}_L(f)$ (or even on the very best elements therein), then the success probability $g_L(\theta)$ can be much larger than $\rho_L(\theta)$, and the effective sample complexity drops. In the extreme, if $\Pr[R_f(A) = 1]$ is bounded below by a constant under $D_{\hat{\theta}}$, then $\mathbb{E}[R_f(A_{\mathrm{out}})]$ becomes $O(1)$ with $K = O(1)$, irrespective of η.

(ii) A stronger generator guarantee than top-L quantiles. Guarantees of the form "D_θ is close (in total variation, KL, or some coupling sense) to the Gibbs distribution $\propto \exp(\beta S_\theta)$" can yield exponential concentration on the highest proxy scores, enabling effective search within $\mathrm{Top}_L(\theta)$ without requiring L to inflate linearly with K. Similarly, if the generator can target $\mathrm{Top}_{cK}(\theta)$ for a small constant c (i.e. η close to 1), the obstruction disappears.

(iii) Additional structure linking proxy scores to the objective. Any assumption that rules out the "proxy-indistinguishable decoys" construction defeats the lower bound. Examples include Lipschitz or margin conditions relating $S_\theta(A)$ to $f(A)$, or parametric models in which the ordering induced by S_θ is stochastically aligned with that of f. Under such conditions, proxy-based filtering effectively increases p with L, and one can realize better-than-η behavior because the top-L region becomes intrinsically easier than unstructured search.

In summary, the $\Omega(\eta)$ lower bound characterizes the price of operating with only a quantile generator and an expensive objective in an otherwise unstructured space: without further assumptions, linear dependence on the inflation factor is the best possible.

8 Experimental plan: evaluating ApproxRoBoT-GEN under no-enumeration protocols

Our experimental objective is twofold: (i) to test whether the qualitative dependencies predicted by the analysis (notably the linear degradation in the inflation factor η and the role of the success probability p) are visible in controlled settings, and (ii) to demonstrate that the proposed exploration–exploitation scheme remains competitive when instantiated with realistic training-free metrics and practical generators.

No-enumeration protocol via large finite pools. Since $\mathcal{A}$ is implicit in our model, we approximate it by a large finite pool $\mathcal{P} \subset \mathcal{A}$ sampled once at the beginning of an experiment (e.g. $|\mathcal{P}| \in \{10^5, 10^6\}$ for DARTS-like cell spaces). All ranks R_f and R_{S_θ} are understood with respect to $\mathcal{P}$, but we enforce a strict access protocol: the algorithm may only obtain an architecture A by calling a generator $G(\theta)$ which returns a sample from an induced distribution D_θ over $\mathcal{P}$, and may only learn $f(A)$ by spending one unit of objective budget. This protocol prevents trivial enumeration-based maximization of S_θ or f, while permitting post-hoc analysis (by the experimenter) of quantities such as $\rho_L(\theta)$ and the realized δ of a generator.

To ensure that $\mathrm{Top}_L(\theta)$ is well-defined despite ties, we fix a deterministic tie-breaking rule (e.g. a hash of the architecture encoding). In all experiments, we log the queried set $\{(A_t, f(A_t))\}_{t=1}^T$ and return $A_{\text{out}} = \arg\max f(A_t)$, matching the algorithmic interface.

Search spaces and objectives. We consider two complementary regimes.

1. *DARTS-like cell spaces.* We take a standard cell-based DAG encoding with a finite operator set and edge choices. The expensive objective $f(A)$ is the validation accuracy after a prescribed training recipe (full training or a consistent low-fidelity proxy such as few-epoch training, with noise controlled by fixed seeds). Training-free metrics $M_i(A)$ are computed without weight training (e.g. SynFlow, Jacobian-based scores, NTK condition surrogates, parameter count, FLOPs).

2. *Implicit adapter/routing spaces.* We define $\mathcal{A}$ as configurations of adapter modules or routing policies in a fixed pretrained backbone (e.g. per-layer adapter rank, placement, and gating). Here $f(A)$ is downstream performance after a fine-tuning budget, while training-free metrics include parameter-efficiency terms, activation statistics from a small calibration set, and gradient-based saliency at initialization of adapters. This regime stresses the "implicit" aspect: the combinatorial space is too large to enumerate, and sampling is the natural mode of access.

Generator implementations with controllable quality. A central goal is to vary generator quality while keeping the rest of the pipeline fixed. We implement a family of generators indexed by a computational effort parameter B and a noise parameter τ:

$$G_{B,\tau}(\theta): \quad \text{sample } B \text{ candidates } \{A^{(b)}\}_{b=1}^{B} \text{ i.i.d. from a base sampler over } \mathcal{P},$$

compute noisy proxy scores $\tilde{S}_\theta(A^{(b)}) = S_\theta(A^{(b)}) + \xi^{(b)}$ with $\xi^{(b)} \sim \mathcal{N}(0, \tau^2)$, and return $A = \arg\max_b \tilde{S}_\theta(A^{(b)})$. As B increases and τ decreases, the returned sample concentrates on higher proxy quantiles, yielding an empirical (L, δ)-quantile guarantee for smaller L (or smaller $\eta = L/K$) at fixed K. In addition, we include an oracle generator baseline $G^\star(\theta)$ that samples uniformly from $\mathrm{Top}_L(\theta)$ (computed offline from $\mathcal{P}$ but never revealed to the algorithm) to isolate the effect of δ from other practical imperfections.

For each run, we estimate δ empirically as the fraction of generator outputs that fall outside $\mathrm{Top}_L(\theta)$ (with θ fixed), and we measure how δ varies with (B, τ). This provides a concrete knob for studying the dependence of performance on generator approximation quality.

Learning θ and defining exploration feedback. We implement the exploration stage in two variants.

1. *Rank-threshold feedback (closest to theory).* Fix $L = \eta K$ and define $Z(A) = \mathbf{1}\{R_f(A) \leq L\}$. During exploration we observe $Z(A_t)$ (possibly with added label noise to simulate stochasticity), and we run GP-UCB over $\theta \in [-1, 1]^M$ to maximize $g_L(\theta) = \mathbb{E}_{A \sim D_\theta}[Z(A)]$.

2. *Direct objective feedback (practical).* We directly use $y_t = f(A_t)$ as the exploration reward and run a BO method over θ (e.g. GP-UCB or Thompson sampling), treating the stochasticity induced by $A \sim D_\theta$ as observation noise. This variant tests whether the rank-threshold abstraction is necessary in practice.

In both cases, we set $\hat{\theta}$ to the best observed θ in exploration (by the corresponding reward), and then perform exploitation by sampling from $G(\hat{\theta})$ for K objective queries. We also include the stronger exploitation variant in which, for each expensive evaluation, we draw $B' \gg 1$ candidates from $G(\hat{\theta})$, rank them by $S_{\hat{\theta}}$, and evaluate only the top candidate, thereby converting proxy computation into fewer objective calls.

Ablations and evaluation metrics. We carry out the following ablations, each repeated over multiple random seeds and multiple independently drawn pools $\mathcal{P}$:

- *Inflation factor and top-set size:* vary η via $L = \eta K$ and report performance as a function of η at fixed K, as well as at fixed total budget $T = T_0 + K$.

- *Generator quality:* vary (B, τ) (and/or switch between $G_{B,\tau}$ and $G^\star$), and report empirical $(\delta, \rho_L(\hat{\theta}), g_L(\hat{\theta}))$ alongside final rank.
- *Exploration–exploitation split:* sweep $T_0 \in \{0, \lfloor T/4 \rfloor, \lfloor T/2 \rfloor, \lfloor 3T/4 \rfloor\}$ with $K = T - T_0$, to test whether performance follows the anticipated tradeoff between learning θ and sampling under $\hat{\theta}$.
- *Proxy set and combiner family:* vary the proxy set size M and compare linear S_θ to constrained variants (e.g. sparse θ, nonnegative θ, or normalized $\|\theta\|_1 = 1$).

Primary outcomes are (i) the achieved best objective value $f(A_{\text{out}})$ and its induced rank $R_f(A_{\text{out}})$ in $\mathcal{P}$, and (ii) empirical curves of $\mathbb{E}[R_f(A_{\text{out}})]$ versus (K, η). Secondary outcomes include estimates of $\rho_L(\theta)$ (precision@L), calibration plots comparing $g_L(\theta)$ to $\rho_L(\theta)$ over explored θ, and the realized cost profile in proxy computations versus objective calls.

Baselines. We compare against (a) random search under the same objective budget; (b) proxy-only selection that fixes θ a priori (e.g. single best metric) and exploits via $G(\theta)$; and (c) objective-only BO over architectures when feasible (restricted to the same sampling access pattern), to clarify the benefit of learning in θ-space coupled to a generator.

Together, these experiments are designed to test not merely raw accuracy but the structural predictions of the framework: how performance varies with η, how generator approximation quality manifests through (δ, ρ_L, g_L), and how exploration in θ mediates the effectiveness of exploitation under a fixed expensive-query budget.

9 Discussion and limitations

On the realism of the (L, δ)-quantile generator assumption. Our analysis isolates a single property of the generator, namely that it returns proxy-high candidates with probability at least $1 - \delta$. This abstraction is intentionally agnostic to the internal mechanism of $G(\theta)$, and it is therefore simultaneously a strength and a limitation. On the one hand, it matches the operational reality of many "generate–score–select" pipelines: if the generator is implemented by drawing a finite batch and returning the best under (possibly noisy) S_θ, then one expects δ to decrease as the internal effort increases. On the other hand, δ may depend strongly on θ, on the presence of proxy ties, and on representational constraints of the sampler (e.g. mode collapse or restricted support), so that a uniform bound $\Pr[A \in \mathrm{Top}_L(\theta)] \geq 1 - \delta$ for all θ can be unrealistic. A more faithful statement is a *local* quantile guarantee on the subset of θ-values visited by the algorithm, or even an *average* guarantee in which δ is replaced by $\mathbb{E}[\mathbf{1}\{A \notin \mathrm{Top}_L(\theta)\} \mid \theta]$. Since all end-to-end bounds enter through the success probability $p = g_L(\theta)$, we may interpret such deviations as a direct deterioration of p, rather than as a failure of the framework. Nevertheless, if $G(\theta)$ sometimes returns candidates far outside proxy top sets, the method can reduce to noisy random search; thus, in practice one should instrument G to report diagnostics (empirical δ, diversity, and proxy-score quantiles) and treat generator tuning as a first-class component.

Relaxing conditional uniformity inside the proxy top set. Theorem 2 uses a conditional-uniformity hypothesis to make the order-statistics calculation explicit. This assumption is not essential for the qualitative message (improvement with K and degradation with L), but it does determine the exact dependence on L. In the absence of uniformity, the correct quantity is the conditional mass that D_θ assigns to the intersection $\mathrm{Top}_L(\theta) \cap \mathrm{Top}_L(f)$; Theorem 1 already exposes this via $\underline{p}_L(\theta)$. One may therefore replace the uniform-rank model by a *domination* condition of the form

$$\Pr\bigl(R_f(A) \le r \,\big|\, A \in \mathrm{Top}_L(\theta) \cap \mathrm{Top}_L(f)\bigr) \;\ge\; \frac{r}{L} \qquad \forall r \in \{1, \dots, L\},$$

which asserts that, conditional on being in the true top-L, the generator is at least as biased toward smaller ranks as a uniform draw. Under such a condition the expected best rank can only improve relative to Theorem 2. Conversely, if the generator concentrates on a small subset of the intersection (e.g. repeatedly returning near-duplicates), then the effective number of distinct "chances" is smaller than K, and the order-statistics benefit can disappear. This suggests a practical modification: encourage high entropy or explicit diversity in D_θ restricted to $\mathrm{Top}_L(\theta)$ (e.g. via rejection sampling with novelty constraints), thereby making the conditional-uniformity approximation more accurate and, empirically, increasing the "effective" X in the binomial model.

Multi-objective constraints and feasibility. Many NAS settings impose hard constraints (latency, memory, energy) or treat them as additional objectives. Our framework can accommodate such requirements, but the appropriate notion of "top-L" must be chosen with care. For hard constraints defining a feasible set $\mathcal{C} \subseteq \mathcal{A}$, one may simply restrict attention to $\mathcal{C}$ by defining $f_{\mathcal{C}}(A) = f(A)$ for $A \in \mathcal{C}$ and $f_{\mathcal{C}}(A) = -\infty$ otherwise, and similarly compute $\mathrm{Top}_L(\theta)$ with respect to proxy scores on $\mathcal{C}$. The resulting guarantees then apply to the best feasible design found. For genuine multi-objective optimization, one may replace scalar ranks R_f by Pareto ranks or by scalarizations f_λ with user-chosen tradeoff λ; the latter fits our notation directly and can be explored by augmenting θ (or coupling θ with λ) at the cost of a larger search domain. An open technical question is whether one can obtain bounds directly in terms of Pareto-optimality notions without committing to a scalarization, since the probability of hitting the Pareto front may be small even when scalarized optima are accessible.

Nonstationary or context-dependent objectives. The model assumes a fixed objective f, yet in practice the evaluation protocol can drift (training recipe changes, data distribution shifts, or stochasticity with heavy tails). If the objective is nonstationary across rounds, $g_L(\theta)$ becomes time-dependent, and the exploration stage is more accurately modeled as an online learning problem with drifting rewards. Standard remedies (sliding-window or discounted GP-UCB, change-point detection, or restarting) can be incorporated, but any guarantee must then degrade with a variation budget measuring how fast f_t changes. A related issue is context dependence: one may wish to select architectures specialized to a context c (device, batch size, dataset slice), in which case $f(A; c)$ varies with c. Here one can treat c as an additional input to the surrogate over θ (learning $g_L(\theta; c)$), but this again increases sample complexity and highlights the importance of cheap, informative proxies.

Implications for proxy design and calibration. Our bounds make explicit that performance hinges not on correlation between $S_\theta(A)$ and $f(A)$ globally, but on $\rho_L(\theta)$, i.e. precision within the proxy top set. This points toward a design principle for training-free metrics: proxies should be constructed to maximize the attainable $\max_\theta \rho_L(\theta)$ for the relevant L, and to make $\rho_L(\theta)$ sufficiently *learnable* from stochastic samples. In particular, proxies that only separate mediocre architectures may have high global correlation yet poor ρ_L at small L, which is precisely the regime exploited by budgeted search. Moreover, Theorem 4 appeals to a calibration relation between $g_L(\theta)$ (the learnable success probability) and $\rho_L(\theta)$ (the quantity appearing in the success lower bound); this calibration need not hold automatically. Consequently, a practical proxy suite should be validated not only by marginal statistics (e.g. Spearman correlation) but also by *top-set calibration plots* that compare estimated $g_L(\theta)$ to empirically measured $\rho_L(\theta)$ over a range of θ. Finally, proxy normalization and robustness matter: since S_θ is linear, heavy-tailed or poorly scaled M_i can make optimization over θ ill-conditioned, thereby increasing the effective exploration budget T_0 required to identify a good combiner.

Taken together, these limitations delineate the boundary of what can be guaranteed under generator-only access. They also suggest concrete research directions: designing generators with controllable δ and diversity, deriving exploitation bounds under weaker (and verifiable) distributional assumptions than conditional uniformity, and developing proxy families whose top-set precision is both high and stably learnable from few expensive evaluations.

10 Conclusion

We studied a training-free NAS setting in which the architecture space $\mathcal{A}$ is accessed only implicitly: we cannot enumerate candidates, and we cannot compute $A(\theta) = \arg\max_A S_\theta(A)$ even for a fixed proxy S_θ. Instead, the algorithm interacts with $\mathcal{A}$ through two oracles: a candidate generator $G(\theta)$ producing draws $A \sim D_\theta$, and an expensive objective oracle $f(A)$. Within this access model, we proposed an explicit separation of concerns: (i) *proxy combination* via a parametric score $S_\theta(A) = \sum_i \theta_i M_i(A)$ built from training-free metrics M_i, and (ii) *approximate maximization* via a quantile generator that returns proxy-high candidates with high probability. This abstraction matches a large class of practical "generate–score–select" pipelines while making the approximation induced by the generator legible in the analysis.

Our main technical contribution is an end-to-end guarantee stated directly in terms of the returned architecture's *true rank* $R_f(A_{\text{out}})$. The analysis rests on two quantities that are intrinsic to the implicit-search setting: the proxy top-set $\text{Top}_L(\theta)$ and its precision against the true objective, $\rho_L(\theta)$. In particular, rather than appealing to global correlation between S_θ and f, the theory identifies $\rho_L(\theta)$ as the operative measure: the algorithm only benefits from a proxy insofar as it concentrates probability mass on truly good candidates within the region that the generator can reach. Given a generator guarantee of the form $\Pr_{A\sim D_\theta}[A \in \text{Top}_L(\theta)] \geq 1 - \delta$, we obtain a lower bound on the probability $g_L(\theta)$ of sampling a truly top-L architecture, and therefore an explicit expression for the expected best rank after K exploitation queries. When one sets $L = \eta K$, the resulting bound exposes a clean η-dependence: approximate maximization that inflates the proxy top-set by a factor η yields a corresponding degradation in expected true rank that is essentially linear in η, up to the success-probability term. Complementing this, we proved a worst-case lower bound showing that $\Omega(\eta)$ dependence is unavoidable for any method constrained to generator-only access, even if the generator were perfect in the sense of sampling uniformly from $\text{Top}_L(\theta)$. Thus, within the adopted access model, the analysis is not merely sufficient but qualitatively sharp.

A conceptual message is that "training-free NAS with guarantees" is possible in implicit spaces, but only after committing to the correct interface between optimization and generation. Theorems phrased in terms of $g_L(\theta)$, $\rho_L(\theta)$, and δ isolate exactly what must be measured or improved to obtain better empirical performance: one may improve the proxy suite (increase attainable ρ_L), improve the generator (decrease δ and increase diversity within proxy-high regions), or increase exploitation budget K. Moreover, the exploration–exploitation decomposition clarifies where expensive evaluations are spent: exploration is used to find a parameter $\hat{\theta}$ with large success probability, while exploitation converts this probability into a best-of-K guarantee through order statistics.

Several open problems remain. First, our model of learnability assumes access (directly or indirectly) to rank-threshold feedback, which is analytically convenient but not always operationally available. A principled theory for learning θ from real-valued $f(A)$ under the two-stage sampling process $A \sim D_\theta$, possibly with heavy-tailed noise and heteroscedasticity induced by the generator, would substantially broaden applicability. Second, the quantile generator assumption treats D_θ as exogenous, whereas in practice $G(\theta)$ is itself trained, tuned, or adapted. An appealing direction is a joint theory in which the generator is an agent with its own sample complexity and failure modes, so that δ and the conditional mass on $\mathrm{Top}_L(\theta) \cap \mathrm{Top}_L(f)$ become controllable quantities rather than fixed constants.

Third, the exploitation analysis is most transparent under conditional-uniformity (or related domination conditions). Developing verifiable conditions under which weaker diversity assumptions suffice—for example, bounds in terms of an effective sample size or a mixing time of a Markov-chain generator restricted to $\mathrm{Top}_L(\theta)$—would help bridge theory to modern samplers that are neither uniform nor i.i.d. Relatedly, it is natural to allow the proxy top-size L to be adaptive: early exploitation could use larger L (higher coverage) and later exploitation smaller L (higher precision), suggesting an η-schedule optimized for the remaining budget and empirically observed success rates. A corresponding analysis would connect our static $L = \eta K$ choice to sequential design.

Fourth, practical NAS pipelines often include additional costs beyond objective queries: proxy computation time, generator sampling time, and wall-clock constraints that depend on architecture size. An explicit multi-budget theory—in which the algorithm chooses how many generator samples to draw per objective evaluation, and where rejection/selection is itself costly—would yield guidance for realistic settings where compute is dominated by generation or by proxy evaluation. Finally, transferring information across runs (datasets, devices, training recipes) is largely unaddressed in our framework. If θ (or the proxy suite) can be warm-started from prior tasks, then the exploration budget T_0 may be amortized, but doing so rigorously requires a theory of task similarity for $\rho_L(\theta)$ or $g_L(\theta)$ under distribution shift.

In summary, the present work formalizes a tractable interface for training-free NAS in implicit architecture spaces and proves rank-based guarantees that make approximation costs explicit. The framework highlights a small set of measurable quantities—top-set precision, generator quantile quality, and effective diversity—that determine performance. We view the main value of these guarantees not as a claim of universal optimality, but as a scaffold on which richer models of generators, proxies, and evaluation protocols can be built while preserving end-to-end, budget-aware statements about the quality of the architecture ultimately returned.