

Article

Budgeted Fixed-Proxy Search for Zero-Shot NAS via Evaluation-Efficient Memetic NAS

Seungyeop Kang and Wangduk Seo * 

Division of AI Computer Science and Engineering, Kyonggi University, Suwon 16227, Republic of Korea; rkdtmdduq137@kyonggi.ac.kr

* Correspondence: wdseo@kyonggi.ac.kr

Abstract

Zero-cost proxies reduce the cost of neural architecture search (NAS), the automated design of neural network architectures, by scoring untrained candidate architectures without full training. Although this makes individual evaluations inexpensive, zero-shot NAS still operates over a large architecture space and therefore depends not only on proxy quality but also on how proxy evaluations are allocated under a limited budget. Despite this role, many zero-shot NAS studies have primarily focused on proxy design, while search over large architecture spaces has often relied on standard evolutionary procedures with limited attention to fixed-proxy budget allocation. In this study, we propose Evaluation-Efficient Memetic NAS (EEM-NAS), a memetic framework for budgeted fixed-proxy zero-shot NAS. EEM-NAS combines bounded one-flip local search, rank-plus-distance target selection, refined-candidate reinsertion, and entropy-guided mutation within one fixed-proxy search process. These components exploit promising candidates, reduce redundant refinement, preserve useful local improvements, and maintain population-level exploration under the same proxy-evaluation budget. On NAS-Bench-201 TSS with eight zero-cost proxies, three datasets, and four fixed-proxy baselines under a budget of 100 proxy evaluations per run, EEM-NAS records 93 wins, three ties, and zero losses over the 96 proxy–dataset–baseline comparisons on final proxy score under paired statistical testing. Because a higher proxy score is informative of final accuracy only when the proxy ranks architectures well, we also report the macro-average test accuracy of the top-10 architectures found in each run, where EEM-NAS achieves the highest macro-average top-10 accuracy among the compared methods on the three most rank-aligned proxies.

Keywords: zero-shot neural architecture search; memetic algorithm; evolutionary algorithm; zero-cost proxy; local search; NAS-Bench-201



Academic Editors: Chien-Chang Chen, Chinchih Chang and Sean Hsiao

Received: 23 June 2026

Revised: 24 July 2026

Accepted: 30 July 2026

Published: 1 August 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Neural architecture search (NAS) automates the design of neural network architectures and has been applied to image classification, object detection, semantic segmentation, and medical image analysis [1–3]. Recent rehabilitation applications still rely on specialized architectures for indoor object detection and non-contact gesture recognition [4,5], illustrating the design effort that NAS can reduce. Conventional NAS often requires full or partial training of many candidate architectures, making architecture evaluation computationally expensive [6]. To alleviate this computational burden, zero-shot NAS replaces training-based evaluation with zero-cost proxies that score untrained networks using initialization-time signals derived from activations, gradients, or Jacobians [7–9].

This shift makes architecture evaluation much cheaper and is particularly attractive in resource-constrained settings, where architectures may need to be adapted quickly to new tasks or hardware variants [10]. Nevertheless, zero-shot NAS still operates over large architecture spaces, so a practical search procedure must decide which architectures to evaluate under a finite proxy-query budget.

Because exhaustive proxy evaluation is impractical in large architecture spaces, zero-shot NAS commonly relies on search procedures that allocate proxy evaluations to a small subset of candidate architectures. Evolutionary search is a natural choice for this setting because it explores discrete architecture spaces through population-based selection and mutation without requiring gradient information from the search objective [11]. In zero-shot NAS, architecture search has typically relied on such general-purpose procedures, while most research effort has been directed at proxy design rather than the search procedure itself [8]. Nevertheless, mutation-driven exploration alone may allocate evaluations across broad structural neighborhoods without systematically exploiting nearby high-scoring candidates. Local search can strengthen exploitation by refining promising candidates through small structural modifications [12–14]. Repeated refinement of top-ranked but structurally similar candidates can waste evaluations on overlapping local neighborhoods. At the same time, stronger local refinement can accelerate convergence toward a narrow set of structural patterns, reducing population diversity and weakening subsequent mutation-driven exploration. These two issues suggest that zero-shot NAS requires more than adding local search to an evolutionary search process.

Achieving both exploration and refinement becomes more difficult as the proxy-evaluation budget decreases. Under a fixed budget, the effect of each allocation decision becomes larger because poorly used evaluations cannot easily be recovered later in the search. Such a budget constraint arises when proxy scores must be computed for each queried architecture rather than retrieved from a precomputed benchmark [15]. In this study, we examine this allocation problem on a fixed benchmark by setting a small number of proxy evaluations per run. Under this setting, performance differences primarily reflect how each search procedure allocates the same proxy-evaluation budget.

To address these challenges, we propose Evaluation-Efficient Memetic NAS (EEM-NAS), a memetic framework for budgeted fixed-proxy zero-shot NAS. EEM-NAS first selects local-refinement targets through rank-plus-distance target selection, which chooses high-scoring candidates that are structurally distinct from previously selected targets. The selected candidates are then refined by bounded one-flip local search within their immediate Hamming neighborhoods. After refinement, the refined candidates are reinserted into the population through a windowed replacement rule based on Restricted Tournament Selection (RTS) so that local improvements influence subsequent population updates [16]. Meanwhile, entropy-guided mutation maintains population-level exploration by perturbing edges whose operation choices have become concentrated in the current population.

The contributions of this study are as follows.

- We formulate low-budget zero-shot NAS as a budgeted fixed-proxy search problem. Rather than modifying or combining proxy signals, we keep a single proxy fixed throughout each run and study how a limited proxy-evaluation budget should be allocated.
- We propose EEM-NAS, a memetic framework in which local refinement is coupled to the evaluation budget by design. Every refinement decision, including which candidates to refine, which neighbors to inspect, and which refined candidates to retain, is made and charged under one fixed proxy-evaluation budget, whereas prior memetic NAS relies on training-based performance estimation.

- We introduce entropy-guided mutation, which maintains structural diversity using operation statistics that are already available in the current population and therefore adds no proxy evaluations. This contrasts with uniform mutation, which applies the same pressure to every edge regardless of the population state.

The remainder of this paper reviews the related work (Section 2), presents EEM-NAS (Section 3), reports the experiments (Section 4), discusses the limitations (Section 5), and concludes the study (Section 6).

2. Related Work

2.1. Zero-Cost Proxies and Budgeted Search

Representative zero-cost proxies exploit diverse signals derived from untrained networks. For example, NASWOT and Jacov use activation-pattern statistics and input-Jacobian correlations to estimate representation diversity and input sensitivity [7,8]. Gradient-based proxies, including Grad Norm, SNIP, SynFlow, and ZiCo, evaluate parameter importance and gradient-flow characteristics without training [8,9,17,18]. Fisher-information-based approaches further estimate architecture quality through activation-gradient saliency aggregation [19,20]. Together, these proxies provide diverse scoring signals for zero-shot NAS. The zero-cost proxies evaluated in this study are summarized in Table 1.

Table 1. Zero-cost proxies evaluated in this study.

Proxy	Family	Search-Role Summary	Reference
ZiCo	gradient statistics	Layer-wise inverse coefficient-of-variation score from gradients	[9]
NWOT	activation patterns	Log-determinant of an untrained activation-pattern kernel	[7]
SynFlow	gradient flow	Data-free saliency propagated through a linearized network	[17]
Jacov	input Jacobian	Spectral aggregation of input-gradient correlation	[8]
SNIP	gradient saliency	First-order parameter saliency from pruning	[18]
Grad Norm	gradient magnitude	Sum of layer-wise gradient norms	[8]
Fisher	Fisher information	Squared activation-gradient saliency aggregated to a network score	[19,20]
SWAP	activation patterns	Expressivity measured by sample-wise activation patterns over an input batch	[21]

Recent studies introduce new single-proxy scores such as SWAP and MeCo [21,22], or combine multiple signals through fusion, ensembles, and entropy- or topology-based criteria [23–25]. Their primary focus remains the scoring signal rather than fixed-budget search allocation. Proxy quality is commonly assessed by exhaustive correlation analysis over a tabulated benchmark, and architecture search, when performed, is carried out with general-purpose procedures such as random sampling or regularized evolution [8,11], without specific regard to how a fixed proxy-evaluation budget is allocated. These proxy-level approaches modify or augment the scoring signal itself, whereas the present work keeps the proxy fixed and instead focuses on optimizing the search procedure under a fixed evaluation budget.

2.2. Memetic Algorithms and Local Search

Memetic algorithms (MAs) extend population-based evolutionary search by coupling local refinement procedures with the evolutionary process [12,13]. Locally refined solutions can either replace existing population members or influence search without modifying the

underlying genotype, corresponding to the classical distinction between Lamarckian and Baldwinian learning schemes [26]. In Lamarckian schemes, the genotype itself is replaced by the locally improved solution, allowing refined structures to be directly inherited by future generations. By contrast, Baldwinian schemes use local refinement only to influence fitness evaluation while preserving the original genotype in the population. This distinction highlights the tension between persistent refinement accumulation and preservation of broader population diversity [27,28].

Recent memetic studies have explored adaptive local-search control and diversity-preservation mechanisms to reduce premature convergence and have examined how local search reshapes population diversity in combinatorial and swarm-based settings [29,30]. Other work has further discussed broader strategies for integrating local refinement within evolutionary search across complex combinatorial problems [31]. Nevertheless, coordinating refinement gains with broad structural exploration under strict evaluation-budget constraints has received relatively limited attention in zero-shot NAS, where the search process operates under a fixed proxy-evaluation budget.

Local refinement has also been combined with training-free evaluation in prior NAS research. A representative cell-based memetic NAS method couples evolutionary search with local refinement under one-epoch training-based performance estimation [14]. Pareto local search uses a training-free proxy to improve multi-objective approximation fronts that trade off accuracy and complexity [32]. Training-free proxies have also been used to guide local-search exploration before subsequent architecture selection using successive halving and training-based validation accuracy [33]. Unlike prior training-based memetic NAS, EEM-NAS performs no architecture training, optimizes one fixed proxy, and charges both offspring and local neighbors to the same FFC budget. Its distinction is this budget-coupled refinement and Lamarckian retention, not local search alone.

3. Evaluation-Efficient Memetic NAS

3.1. Preliminaries

The search space considered in this study is the topology search space (TSS) of NAS-Bench-201 [34]. Each architecture is represented as a directed acyclic graph with four ordered nodes, where node 0 is the cell input, nodes 1 and 2 are intermediate feature nodes, and node 3 is the cell output, as illustrated in Figure 1a. The six directed edges connecting every earlier-to-later node pair are each independently assigned one operation from $\mathcal{O}$ (Figure 1b):

$$\mathcal{O} = \{\text{none}, \text{skip_connect}, \text{nor_conv_1x1}, \text{nor_conv_3x3}, \text{avg_pool_3x3}\}, \quad |\mathcal{O}| = 5.$$

An architecture is therefore encoded as a length- L operation vector

$$a = (a_{01}, a_{02}, a_{12}, a_{03}, a_{13}, a_{23}) \in \mathcal{O}^L, \quad L = 6,$$

yielding a total search space of $|\mathcal{O}|^L = 5^6 = 15,625$ architectures, as illustrated in Figure 1c.

Search cost is measured in fitness function calls (FFCs). A fitness function call is counted each time the search process queries the proxy ψ on an architecture, including both reproduction offspring and local-search neighbors. The total per-run budget is denoted by B_{FFC} , and the search terminates once this budget is exhausted. All search decisions rely solely on proxy evaluations without using downstream test accuracy during search.

NAS-Bench-201 TSS Search Space and Encoding

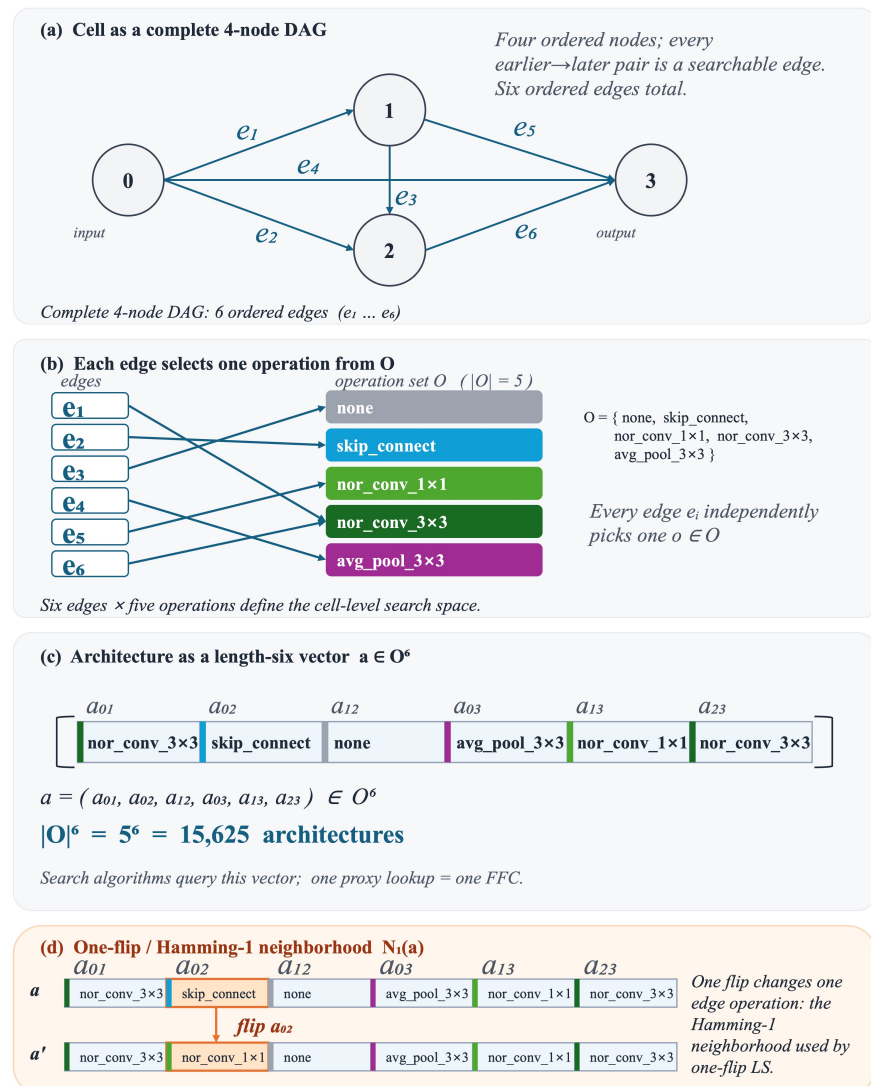


Figure 1. NAS-Bench-201 TSS cell encoding used in this study. The searchable cell is a complete four-node DAG with six ordered edges, and each edge selects one operation from a five-operation set. The resulting vector $a = (a_{01}, a_{02}, a_{12}, a_{03}, a_{13}, a_{23})$ is the state manipulated by all search procedures. The same encoding induces the Hamming-1 local-search neighborhood.

Local refinement operates on the Hamming neighborhood induced by this encoding. The Hamming distance between two architectures $a, b \in \mathcal{O}^L$ is the number of edge positions on which their operation assignments differ,

$$d_H(a, b) = \sum_{e=1}^L \mathbf{1}[a_e \neq b_e],$$

and the Hamming-1 neighborhood of an architecture a is the set of all architectures that differ from a on exactly one edge,

$$\mathcal{N}_1(a) = \{b \in \mathcal{O}^L : d_H(a, b) = 1\}.$$

Because each of the $L = 6$ edges can be changed to any of the remaining $|\mathcal{O}| - 1 = 4$ operations, the neighborhood size is fixed,

$$|\mathcal{N}_1(a)| = L(|\mathcal{O}| - 1) = 6 \times 4 = 24,$$

and Figure 1d illustrates a Hamming-1 move in which exactly one edge operation changes. Given a fixed zero-cost proxy ψ and an FFC budget B_{FFC} , the objective is to identify a high-proxy-score architecture using at most B_{FFC} proxy evaluations.

3.2. Method Overview

Figure 2 summarizes the overall EEM-NAS search process. EEM-NAS combines evolutionary search with periodic local refinement under a shared proxy-evaluation budget. During reproduction, offspring are generated through tournament selection, uniform crossover, and entropy-guided mutation. During local refinement, selected architectures are refined within their Hamming-1 neighborhoods. The local-search output is reinserted into the population through rank-based replacement, allowing useful local improvements to influence later population updates. All decisions use only the proxy fixed for the run and the current population state. The search process does not use proxy fusion, learned predictors, supernet training, or test-accuracy feedback.

EEM-NAS Overview

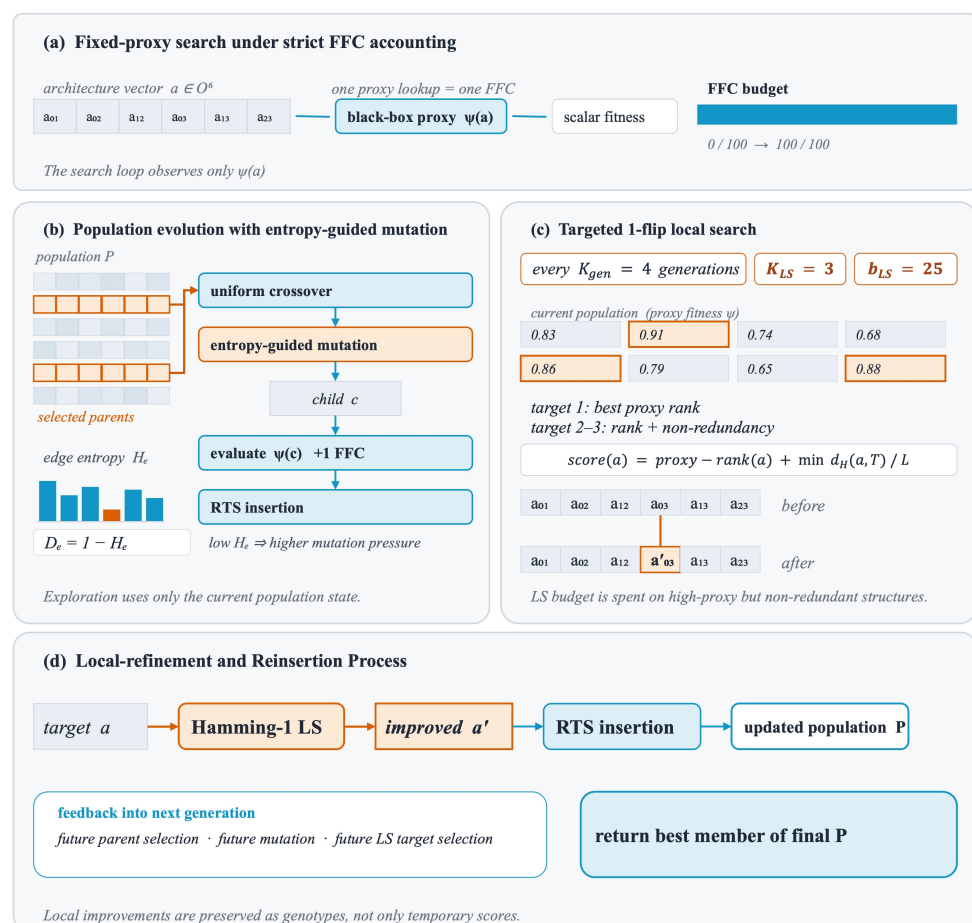


Figure 2. Overview of Evaluation-Efficient Memetic NAS (EEM-NAS). Each run optimizes one fixed zero-cost proxy under a shared FFC budget. EEM-NAS combines population-based reproduction, entropy-guided mutation, rank-plus-distance local-search target selection, bounded Hamming-1 local search, and RTS-based reinsertion of locally improved architectures.

3.3. Evolutionary Search

The population is maintained as a fixed-size collection

$$P = (a_1, \dots, a_N)$$

of N candidate architectures, where each population member a_i is associated with a proxy fitness value $\psi(a_i)$. The initial population is constructed by sampling N architectures uniformly at random from $\mathcal{O}^L$ and evaluating each using the proxy ψ , consuming N FFCs from B_{FFC} . The reproduction step then proceeds as follows (Figure 2b).

Reproduction begins with parent selection and crossover. Crossover is performed with probability p_{cross} by independently selecting two parents using tournament selection of size k , where the highest-scoring architecture among k population members sampled uniformly without replacement from P is chosen as the parent. A child architecture is then generated through edge-wise uniform crossover, where each edge e independently inherits its operation from one of the two parents with equal probability $\frac{1}{2}$. Otherwise, with probability $1 - p_{\text{cross}}$, a single parent is selected by tournament and copied directly without crossover.

The entropy-guided mutation operator mitigates premature structural convergence by preferentially perturbing edges whose operation distributions have become highly concentrated across the population. For each edge e and operation o , let

$$c_{e,o} = \sum_{i=1}^N \mathbf{1}[(a_i)_e = o]$$

denote the number of architectures in the current population assigning operation o to edge e , and let

$$q_{e,o} = \frac{c_{e,o}}{N}$$

be the corresponding population frequency. The normalized Shannon entropy of edge e and the associated entropy deficit are defined as

$$H_e = -\frac{1}{\log |\mathcal{O}|} \sum_{o \in \mathcal{O}} q_{e,o} \log q_{e,o}, \quad D_e = 1 - H_e, \quad 0 \leq H_e \leq 1,$$

under the convention $0 \log 0 = 0$. Edges with lower entropy therefore receive larger deficit values and become more likely to be selected for mutation. To prevent mutation probabilities from collapsing to zero, a small floor $\eta = 0.05$ is added to each entropy deficit,

$$u_e = D_e + \eta, \quad \pi_e = \frac{u_e}{\sum_j u_j},$$

where π_e denotes the normalized probability of selecting edge e for mutation. With base mutation rate $p_{\text{mut}} = 1/L$, each edge is independently selected for mutation with probability

$$\min(0.75, \pi_e L p_{\text{mut}}).$$

The factor $\pi_e L p_{\text{mut}}$ redistributes mutation probability across edges according to entropy while preserving the nominal mutation pressure $L p_{\text{mut}}$ since $\sum_e \pi_e = 1$. Two safeguards modify this base rate. The cap of 0.75 prevents any single edge from receiving an excessively large mutation probability, and, if no edge is selected, one edge is sampled from π so that every offspring undergoes at least one mutation step. Once an edge e is selected for mutation, the same population counts $c_{e,o}$ are reused to choose its new operation. The

replacement operation is sampled from $\mathcal{O} \setminus \{a_e\}$ with probability inversely proportional to its current population frequency,

$$\Pr(a_e \leftarrow o) = \frac{(c_{e,o} + 1)^{-1}}{\sum_{o' \in \mathcal{O} \setminus \{a_e\}} (c_{e,o'} + 1)^{-1}}.$$

This choice favors operations that appear less frequently on the selected edge, with the +1 term keeping the weight finite for operations currently absent from that edge. The mutation rule therefore acts as a diversity-oriented candidate-generation bias, while population replacement remains fitness-driven through RTS.

Each mutated offspring candidate c is then considered for population replacement through the RTS-based windowed replacement rule. Whereas the original RTS compares a candidate against its most similar window member to promote niching among multimodal solutions, here the candidate competes against the lowest-scoring window member, which keeps the rule purely proxy-driven while preserving the restricted sampled-window replacement structure. A window $W \subseteq P$ of size $|W|$ is sampled uniformly without replacement, and the lowest-scoring member

$$p^* = \arg \min_{p \in W} \psi(p)$$

is identified.

If $\psi(c) > \psi(p^*)$, then p^* is replaced by c . Otherwise the population remains unchanged. The same RTS replacement rule is used when reinserting architectures returned by local search. Overall, the evolutionary phase combines entropy-guided mutation with windowed replacement to maintain structural diversity under strict FFC constraints. Entropy-guided mutation increases the probability of perturbing overconcentrated edges, while the windowed replacement rule limits replacement to sampled population windows rather than enforcing fully global replacement.

3.4. Local Search

Every $K_{\text{gen}} = 4$ generations, EEM-NAS performs a bounded local-search pass over a selected subset of the current population (Figure 2c).

Local-search targets are chosen by rank-plus-distance selection. The $K_{\text{LS}} = 3$ local-search targets are selected greedily from P . The first target is chosen as the highest-scoring population member so that local refinement begins from the strongest proxy candidate. Each subsequent target maximizes a combined rank-and-diversity score,

$$s(a) = \text{rank}_{[0,1]}(\psi(a)) + \frac{\min_{b \in T} d_{\text{H}}(a, b)}{L},$$

where T is the set of already selected targets, $\text{rank}_{[0,1]}$ maps higher proxy scores linearly to larger normalized ranks in $[0, 1]$, and $d_{\text{H}}(a, b)$ denotes the Hamming distance between architectures measured by the number of edges assigned different operations. The second term acts as a structural diversity bonus, assigning larger scores to architectures that are more distant from previously selected targets in Hamming space.

Each selected target $a \in T$ is then refined by bounded 1-flip local search within its 24-neighbor Hamming-1 neighborhood $\mathcal{N}_1(a)$, where each neighbor is generated by changing the operation assigned to exactly one edge. Neighbors are visited in a seeded random permutation of all elements of $\mathcal{N}_1(a)$, generated from a seed determined by the current generation g and target rank r . If a better neighbor b satisfying $\psi(b) > \psi(a)$ is encountered, the move is accepted and the sweep restarts from b , allowing multiple improvements to be chained within a single call as long as the budget permits. The local-

search call terminates when the per-call FFC budget $\min(b_{LS}, B_{FFC} - FFC)$ is exhausted or when the full neighborhood of the current target is scanned without finding a strictly better neighbor. Each newly inspected neighbor consumes one FFC, while the starting target is not double-charged within the same call. The per-call budget is set to $b_{LS} = 25$. Because the starting architecture is preseeded into the per-call cache, a no-improvement scan of all 24 neighbors costs exactly 24 new evaluations; the additional unit is available only after an improving move restarts the scan.

The architecture returned by the local-search call is reintroduced into the population through the same windowed replacement rule used for offspring candidates. This reinsertion step allows local-search outcomes to influence subsequent evolutionary search without globally replacing the population. Because both offspring and locally refined candidates are admitted to the population only when their proxy score is strictly higher than that of the lowest-scoring member in the sampled replacement window, the maximum proxy score of the retained population is non-decreasing across replacement attempts. Overall, the local-search phase combines three elements under the shared FFC budget. Rank-plus-distance target selection reduces repeated refinement around structurally similar candidates, bounded 1-flip local search concentrates evaluations on small local modifications, and RTS-based reinsertion allows useful local improvements to persist when they are competitive within the sampled replacement window.

Algorithm 1 presents the complete pseudocode, with Algorithms 2 and 3 detailing the RTS insertion and LS target-selection subroutines.

Algorithm 1 Evaluation-Efficient Memetic NAS (EEM-NAS).

Require: proxy ψ , budget B_{FFC} , population size N , LS period K_{gen} , LS targets K_{LS} , window size $|W|$, LS budget b_{LS}

- 1: Initialize P with N random architectures and evaluate them
- 2: $g \leftarrow 0$
- 3: **while** FFC budget remains **do**
- 4: $g \leftarrow g + 1$
- 5: **for** $i = 1, \dots, N$ **do**
- 6: **if** FFC budget exhausted **then break**
- 7: **end if**
- 8: Select parents by tournament selection
- 9: Produce child c by uniform crossover and entropy-guided mutation
- 10: Evaluate c and call RTS-INSERT($P, c, |W|$)
- 11: **end for**
- 12: **if** $g \bmod K_{gen} = 0$ and FFC budget remains **then**
- 13: $T \leftarrow \text{LS-TARGET-SELECT}(P, K_{LS})$
- 14: **for** $a \in T$ **do**
- 15: **if** FFC budget exhausted **then break**
- 16: **end if**
- 17: $a' \leftarrow \text{first-improvement 1-flip local search from } a \text{ with budget } \min(b_{LS}, B_{FFC} - FFC)$
- 18: RTS-INSERT($P, a', |W|$) ▷ reinsert local-search output
- 19: **end for**
- 20: **end if**
- 21: **end while**
- 22: **return** highest-fitness member of P

Algorithm 2 RTS-INSERT.**Require:** population P , candidate c , window size $|W|$

- 1: Sample $W \subseteq P$ of size $|W|$ uniformly without replacement
- 2: $p^* \leftarrow \arg \min_{p \in W} \psi(p)$
- 3: **if** $\psi(c) > \psi(p^*)$ **then**
- 4: Replace p^* by c
- 5: **end if**

Algorithm 3 LS-TARGET-SELECT.**Require:** population P , target count K

- 1: $T \leftarrow \{\arg \max_{a \in P} \psi(a)\}$
- 2: **while** $|T| < \min(K, N)$ **do**
- 3: Select $a^* = \arg \max_{a \in P \setminus T} [\text{rank}_{[0,1]}(\psi(a)) + \min_{b \in T} d_H(a, b)/L]$
- 4: $T \leftarrow T \cup \{a^*\}$
- 5: **end while**
- 6: **return** T

EEM-NAS is invariant to order-preserving transformations of proxy scores. It uses proxy scores only through pairwise comparisons or induced ranks, while entropy-guided mutation depends on population operation frequencies. Therefore, under a fixed random seed and deterministic tie handling, replacing ψ with $f \circ \psi$ for any strictly increasing function f preserves the search trajectory. This invariance removes sensitivity to order-preserving differences in proxy scale, but it does not correct errors in the proxy-induced architecture ordering.

3.5. Operating Configuration and Complexity

The primary operating configuration employs $N = 10$, $K_{\text{gen}} = 4$, $K_{\text{LS}} = 3$, $|W| = 3$, $b_{\text{LS}} = 25$, and a baseline mutation pressure of $1/L$ per edge before entropy reweighting. The population size $N = 10$ is fixed throughout the main experiments and is not treated as a separate sweep parameter. This setting keeps the initialization cost at 10% of the FFC budget while reserving most of the budget for reproduction and local refinement. The local-search cap $b_{\text{LS}} = 25$ corresponds to one quarter of the total FFC budget (100). Because an NAS-Bench-201 1-flip neighborhood contains $L(|\mathcal{O}| - 1) = 24$ neighbors, a single local-search call can traverse the full neighborhood without exhausting the overall budget. For reproducibility, the tournament size is $k = 5$, and uniform crossover is applied with probability $p_{\text{cross}} = 0.9$. If crossover is not applied, the first selected parent is copied before mutation. The Hamming-1 local-search scan employs a random permutation of the 24 neighbors using the seed $10007g + 131r + 7$, where g denotes the generation and r denotes the local-search target rank.

Duplicate handling follows a deliberately conservative accounting policy. The main evaluator charges one FFC for every proxy lookup, including duplicate architecture queries arising during reproduction or local search. Within a single local-search call, the starting architecture is seeded into a local cache, and repeated neighbors in the same call are not charged twice. Entropy-guided mutation employs an at-least-one-edit fallback so that, if the Bernoulli edge mask is empty, one edge is forced to mutate.

The cost measure adopted in this study is the number of proxy evaluations, which is strictly bounded by the FFC budget. Apart from proxy calls, each reproduction step consists of constant-size tournament selection, uniform crossover, entropy-guided mutation over an $L = 6$ edge encoding, and a replacement-window scan over $|W| = 3$ members. Rank-plus-distance target selection scans only the current population, and entropy-guided mutation uses operation counts already available within that population. Each local-search

call is bounded by b_{LS} proxy evaluations, with the scan restarting from each accepted neighbor. In the tabulated NAS-Bench-201 setting, proxy and test-accuracy values are retrieved by lookup, so measured wall-clock time primarily reflects search-process overhead. In an online setting, each FFC corresponds to one zero-cost proxy evaluation on an untrained network, typically requiring a forward pass and, for gradient-based proxies, one additional backward pass. Under this interpretation, the FFC budget provides an architecture-evaluation cost abstraction that is independent of implementation-specific lookup overhead.

4. Experimental Results

4.1. Proxy Definitions

Each run fixes one zero-cost proxy ψ and maximizes its scalar value under the FFC budget. The proxy is computed from an initialized network $F_a(\cdot; \theta)$ associated with architecture a , optionally using a minibatch $\mathcal{B} = \{(x_i, y_i)\}_{i=1}^m$ and the training loss $\mathcal{L}$ when the proxy is data-dependent. The search process observes only the resulting scalar proxy value $\psi(a)$ and does not use post-training test accuracy during search. All eight proxy arrays are precomputed before search and are used in the benchmark-provided higher-is-better orientation.

ZiCo measures the stability of layer-wise gradients through an inverse coefficient-of-variation statistic [9]. Let $g_{\ell,j}^{(b)}$ be the gradient of parameter j in layer ℓ measured on minibatch or split b , and let S be the number of such splits. Define

$$\mu_{\ell,j} = \frac{1}{S} \sum_{b=1}^S g_{\ell,j}^{(b)}, \quad \sigma_{\ell,j} = \sqrt{\frac{1}{S} \sum_{b=1}^S (g_{\ell,j}^{(b)} - \mu_{\ell,j})^2}, \quad |\overline{g}|_{\ell,j} = \frac{1}{S} \sum_{b=1}^S |g_{\ell,j}^{(b)}|.$$

The score used in this paper follows the inverse-CoV aggregation

$$\psi_{\text{ZiCo}}(a) = \sum_{\ell} \log \left(\sum_{j: \sigma_{\ell,j} > 0} \frac{|\overline{g}|_{\ell,j}}{\sigma_{\ell,j}} \right).$$

Larger values indicate more stable gradient signals across minibatches or splits.

NWOT, also known as NASWOT, uses activation patterns at initialization [7]. For each sample x_i , collect binary ReLU activation indicators into a vector $c_i \in \{0, 1\}^R$. The sample-similarity kernel is

$$K_{ij} = c_i^\top c_j + (1 - c_i)^\top (1 - c_j),$$

which counts neurons whose binary activation state agrees between samples. The proxy score is

$$\psi_{\text{NWOT}}(a) = \log |\det K|.$$

Larger scores correspond to more diverse and better-conditioned activation partitions.

SynFlow is a data-free saliency proxy designed to avoid layer collapse during pruning-style scoring [17]. Let $\bar{\theta} = |\theta|$ be the absolute-value-reparameterized parameter vector and let $\mathbf{1}$ denote an all-ones input. With

$$R(a, \bar{\theta}) = \mathbf{1}^\top F_a(\mathbf{1}; \bar{\theta}),$$

the parameter saliency is

$$s_j = \left| \bar{\theta}_j \frac{\partial R}{\partial \bar{\theta}_j} \right|, \quad \psi_{\text{SynFlow}}(a) = \sum_j s_j.$$

The score therefore measures gradient flow through the untrained architecture without using labels or natural images.

Jacov scores the geometry of the input–output Jacobian [8]. Let

$$J_i = \text{vec} \left(\frac{\partial F_a(x_i; \theta)}{\partial x_i} \right)$$

be the flattened input Jacobian for sample i , and let C be the sample correlation matrix built from $\{J_i\}_{i=1}^m$. If $\{\lambda_r\}$ are the eigenvalues of C , the score is

$$\psi_{\text{Jacov}}(a) = - \sum_r \left[\log(\lambda_r + \epsilon) + \frac{1}{\lambda_r + \epsilon} \right].$$

This penalizes degenerate Jacobian spectra and favors architectures with better-conditioned input sensitivity at initialization.

SNIP is a first-order pruning saliency computed at initialization [18]. Introduce a multiplicative mask m_j on parameter θ_j and evaluate the gradient at $m_j = 1$. The saliency and network score are

$$s_j = \left| \frac{\partial \mathcal{L}}{\partial m_j} \right|_{m=1} = \left| \theta_j \frac{\partial \mathcal{L}}{\partial \theta_j} \right|, \quad \psi_{\text{SNIP}}(a) = \sum_j s_j.$$

The score is large when the initialized network has many parameters whose removal would produce a large first-order loss change.

Grad Norm directly aggregates the magnitude of loss gradients over trainable layers [8]. With W_ℓ denoting the weight tensor of layer ℓ ,

$$\psi_{\text{GradNorm}}(a) = \sum_\ell \|\nabla_{W_\ell} \mathcal{L}\|_F.$$

Larger values indicate stronger gradient response at initialization.

Fisher estimates channel importance from activation-gradient products [19,20]. Let $z_{\ell,k,u}^{(i)}$ be activation channel k at spatial position u in layer ℓ for sample i . A channel saliency is

$$s_{\ell,k} = \frac{1}{2m} \sum_{i=1}^m \left(\sum_u z_{\ell,k,u}^{(i)} \frac{\partial \mathcal{L}}{\partial z_{\ell,k,u}^{(i)}} \right)^2, \quad \psi_{\text{Fisher}}(a) = \sum_{\ell,k} s_{\ell,k}.$$

This score is high when the current activations and loss gradients indicate strong Fisher-style channel sensitivity.

SWAP measures untrained-network expressivity through sample-wise activation patterns [21]. For a minibatch $\{x_i\}_{i=1}^m$, each hooked rectified-linear unit u defines

$$(z_u)_i = \mathbf{1}[a_u(x_i) > 0],$$

and the score counts distinct patterns:

$$\psi_{\text{SWAP}}(a) = |\{z_u : u \in U\}|.$$

We use the official plain score with $m = 64$ and average five Gaussian weight re-initializations.

4.2. Experimental Settings

The experiments use the NAS-Bench-201 TSS encoding introduced in Section 3.1 and illustrated in Figure 1. Architectures are evaluated on CIFAR-10, CIFAR-100, and ImageNet-

16–120. The eight proxies defined in Section 4.1 are used throughout the experiments, namely ZiCo, NWOT, SynFlow, Jacov, SNIP, Grad Norm, Fisher, and SWAP. Raw proxy arrays and test accuracies are obtained from the precomputed NAS-Bench-201 benchmark release, and no architecture is trained during search. The SWAP arrays were computed by the authors for all 15,625 architectures with the official SWAP implementation, as described in Section 4.1. The main comparison comprises three datasets $\times$ eight proxies $\times$ five methods $\times$ 100 paired runs under $\text{FFC} = 100$. The compared methods are the final EEM-NAS configuration, random sampling, regularized evolution, simple mutation, and a standard genetic algorithm (GA). All the population-based baselines share the population size, tournament size, and mutation rate used by EEM-NAS whenever the corresponding operator is defined.

Each search run returns the proxy-maximizing architecture within the final population. For random sampling, which has no persistent population, the returned architecture is the best queried architecture. Under the EEM-NAS strict-better RTS replacement rule, locally refined architectures can remain in the population when accepted by the sampled replacement window. In addition to the returned architecture, all the architectures queried during the run are retained for analysis. Because proxy score and post-training test accuracy are not assumed to be monotonic, this study does not treat test accuracy as the primary search-time outcome. As a secondary diagnostic, the unique architectures queried during a run are ranked by the same optimized proxy, and the mean test accuracy of the top 10 is reported. This diagnostic is reported for ZiCo, SWAP, and NWOT because they show the strongest proxy–test rank alignment among the evaluated proxies in Section 4.5. It is reported descriptively and is never used during search.

Each experimental cell is evaluated over $n = 100$ paired random seeds. The seed count was fixed before the win/tie/loss evaluation, and no seed filtering was applied. We employ paired two-sided Wilcoxon signed-rank tests [35,36], Holm step-down correction [37], and Cliff’s δ with effect-size cutoffs [38]. The 96-cell main comparison family comprises eight proxies $\times$ three datasets $\times$ four baselines. A win indicates $p_{\text{Holm}} < 0.05$ and $\delta > 0$ for EEM-NAS over a baseline, while a loss denotes the mirror case. All the remaining cells are treated as ties.

4.3. Main Comparison with Baselines

Table 2 summarizes the main win/tie/loss comparison for fixed-proxy search under $\text{FFC} = 100$. EEM-NAS records 93 wins, three ties, and zero losses across the 96 Holm-corrected proxy–dataset–baseline comparisons; the ties occur against GA on Jacov and regularized evolution on SWAP. No evaluated baseline achieves a Holm-significant proxy-score improvement over EEM-NAS in any evaluated cell. Cliff’s δ has a median of 0.621 and ranges from 0.038 to 0.996. The median effect is categorized as large under the adopted effect-size thresholds [38].

Table 2. Holm-corrected win/tie/loss result at $\text{FFC} = 100$ (8 proxies $\times$ 3 datasets $\times$ 4 baselines; $n = 100$ paired seeds). A win requires $p_{\text{Holm}} < 0.05$ and Cliff’s $\delta > 0$.

Baseline	Win	Tie	Loss
Random	24	0	0
Reg. Evolution	23	1	0
Simple Mutation	24	0	0
GA	22	2	0
Total	93	3	0

To examine how these aggregate outcomes distribute across individual proxy–dataset combinations, Table 3 provides a detailed cell-level breakdown of the main comparison results. The observed proxy-score advantages remain broadly consistent across most evaluated settings, although the magnitude of improvement varies depending on the proxy and search baseline. Specifically, the largest performance gaps appear on saliency-based proxies with broader score ranges. The three unmarked cells in Table 3 are the two Jacov–GA comparisons and the ImageNet-16-120 SWAP–regularized evolution comparison; their mean differences do not survive correction. Because the proxy is fixed within each comparison, performance differences reflect how each search procedure allocates the same evaluation budget to optimize the same objective. The results show that EEM-NAS uses the fixed proxy budget more effectively than the baselines across proxies, datasets, and search strategies. Whether this proxy-score advantage translates into higher architecture quality depends on how well the optimized proxy ranks architectures, which is examined for the most rank-aligned proxies in Section 4.5.

Table 3. Per-cell final proxy scores at FFC = 100 (mean $\pm$ std; $n = 100$). Bold marks the best mean; ▼ marks a baseline significantly worse than EEM-NAS after Holm correction. The last column gives the mean rank across eight proxies.

Dataset	Method	ZiCo	SWAP	NWOT	SynFlow	
CIFAR-10	EEM-NAS	$1.08 \times 10^3 \pm 6.05 \times 10^0$	$3.05 \times 10^5 \pm 8.72 \times 10^3$	832.135 ± 0.238	114.652 ± 1.330	
	Random	$914.086 \pm 80.371 \blacktriangledown$	$2.90 \times 10^5 \pm 1.48 \times 10^4 \blacktriangledown$	$823.898 \pm 6.162 \blacktriangledown$	$107.220 \pm 7.747 \blacktriangledown$	
	Reg. Evolution	$1.06 \times 10^3 \pm 4.82 \times 10^1 \blacktriangledown$	$2.98 \times 10^5 \pm 1.18 \times 10^4 \blacktriangledown$	$831.778 \pm 1.170 \blacktriangledown$	$112.136 \pm 4.972 \blacktriangledown$	
	Simple Mutation	$1.02 \times 10^3 \pm 5.51 \times 10^1 \blacktriangledown$	$2.99 \times 10^5 \pm 1.45 \times 10^4 \blacktriangledown$	$831.081 \pm 2.611 \blacktriangledown$	$111.183 \pm 5.227 \blacktriangledown$	
	GA	$1.05 \times 10^3 \pm 4.72 \times 10^1 \blacktriangledown$	$2.99 \times 10^5 \pm 1.30 \times 10^4 \blacktriangledown$	$831.615 \pm 1.566 \blacktriangledown$	$113.350 \pm 3.290 \blacktriangledown$	
CIFAR-100	EEM-NAS	$1.09 \times 10^3 \pm 7.41 \times 10^0$	$3.03 \times 10^5 \pm 1.19 \times 10^4$	831.763 ± 0.279	117.032 ± 1.331	
	Random	$916.650 \pm 79.194 \blacktriangledown$	$2.87 \times 10^5 \pm 1.51 \times 10^4 \blacktriangledown$	$823.422 \pm 6.139 \blacktriangledown$	$109.513 \pm 7.736 \blacktriangledown$	
	Reg. Evolution	$1.06 \times 10^3 \pm 3.53 \times 10^1 \blacktriangledown$	$2.98 \times 10^5 \pm 1.46 \times 10^4 \blacktriangledown$	$831.201 \pm 1.929 \blacktriangledown$	$114.471 \pm 5.030 \blacktriangledown$	
	Simple Mutation	$1.02 \times 10^3 \pm 5.62 \times 10^1 \blacktriangledown$	$2.95 \times 10^5 \pm 1.38 \times 10^4 \blacktriangledown$	$830.182 \pm 3.341 \blacktriangledown$	$112.773 \pm 6.100 \blacktriangledown$	
	GA	$1.05 \times 10^3 \pm 4.55 \times 10^1 \blacktriangledown$	$2.99 \times 10^5 \pm 1.32 \times 10^4 \blacktriangledown$	$831.119 \pm 2.124 \blacktriangledown$	$115.696 \pm 3.298 \blacktriangledown$	
ImageNet-16-120	EEM-NAS	$1.09 \times 10^3 \pm 8.66 \times 10^0$	$7.74 \times 10^4 \pm 3.15 \times 10^3$	742.304 ± 0.342	113.297 ± 0.188	
	Random	$917.020 \pm 80.451 \blacktriangledown$	$7.41 \times 10^4 \pm 3.35 \times 10^3 \blacktriangledown$	$733.572 \pm 6.131 \blacktriangledown$	$106.048 \pm 7.120 \blacktriangledown$	
	Reg. Evolution	$1.07 \times 10^3 \pm 3.35 \times 10^1 \blacktriangledown$	$7.62 \times 10^4 \pm 3.18 \times 10^3 \blacktriangledown$	$741.720 \pm 1.993 \blacktriangledown$	$110.672 \pm 4.606 \blacktriangledown$	
	Simple Mutation	$1.02 \times 10^3 \pm 5.65 \times 10^1 \blacktriangledown$	$7.61 \times 10^4 \pm 3.45 \times 10^3 \blacktriangledown$	$740.786 \pm 2.957 \blacktriangledown$	$109.719 \pm 5.163 \blacktriangledown$	
	GA	$1.06 \times 10^3 \pm 4.10 \times 10^1 \blacktriangledown$	$7.62 \times 10^4 \pm 3.10 \times 10^3 \blacktriangledown$	$741.478 \pm 2.476 \blacktriangledown$	$111.778 \pm 3.213 \blacktriangledown$	
Dataset	Method	Jacov	SNIP	Grad Norm	Fisher	Avg. Rank
CIFAR-10	EEM-NAS	-65.025 ± 0.035	$3.50 \times 10^3 \pm 1.17 \times 10^3$	$1.87 \times 10^3 \pm 7.23 \times 10^2$	$1.42 \times 10^3 \pm 7.37 \times 10^2$	1.00
	Random	$-65.046 \pm 0.037 \blacktriangledown$	$711.884 \pm 705.877 \blacktriangledown$	$453.579 \pm 436.364 \blacktriangledown$	$100.309 \pm 319.894 \blacktriangledown$	5.00
	Reg. Evolution	$-65.044 \pm 0.037 \blacktriangledown$	$1.79 \times 10^3 \pm 1.27 \times 10^3 \blacktriangledown$	$1.12 \times 10^3 \pm 6.67 \times 10^2 \blacktriangledown$	$626.961 \pm 758.616 \blacktriangledown$	2.75
	Simple Mutation	$-65.035 \pm 0.035 \blacktriangledown$	$1.18 \times 10^3 \pm 9.06 \times 10^2 \blacktriangledown$	$878.709 \pm 673.429 \blacktriangledown$	$349.222 \pm 592.698 \blacktriangledown$	3.75
	GA	-65.028 ± 0.039	$1.81 \times 10^3 \pm 1.28 \times 10^3 \blacktriangledown$	$1.03 \times 10^3 \pm 6.51 \times 10^2 \blacktriangledown$	$559.724 \pm 706.332 \blacktriangledown$	2.50
CIFAR-100	EEM-NAS	-65.016 ± 0.031	$3.36 \times 10^3 \pm 1.17 \times 10^3$	$2.03 \times 10^3 \pm 7.54 \times 10^2$	$2.20 \times 10^3 \pm 2.23 \times 10^3$	1.00
	Random	$-65.048 \pm 0.034 \blacktriangledown$	$747.611 \pm 738.944 \blacktriangledown$	$446.376 \pm 453.075 \blacktriangledown$	$111.838 \pm 537.365 \blacktriangledown$	5.00
	Reg. Evolution	$-65.038 \pm 0.039 \blacktriangledown$	$1.81 \times 10^3 \pm 1.20 \times 10^3 \blacktriangledown$	$1.13 \times 10^3 \pm 8.12 \times 10^2 \blacktriangledown$	$755.081 \pm 1460.643 \blacktriangledown$	2.50
	Simple Mutation	$-65.031 \pm 0.034 \blacktriangledown$	$1.37 \times 10^3 \pm 1.15 \times 10^3 \blacktriangledown$	$871.763 \pm 659.838 \blacktriangledown$	$400.759 \pm 1031.570 \blacktriangledown$	3.75
	GA	$-65.034 \pm 0.037 \blacktriangledown$	$1.70 \times 10^3 \pm 1.20 \times 10^3 \blacktriangledown$	$1.02 \times 10^3 \pm 7.44 \times 10^2 \blacktriangledown$	$648.384 \pm 1396.030 \blacktriangledown$	2.75
ImageNet-16-120	EEM-NAS	-68.870 ± 0.153	$4.48 \times 10^3 \pm 1.07 \times 10^3$	$3.09 \times 10^3 \pm 8.24 \times 10^2$	$2.95 \times 10^3 \pm 1.20 \times 10^3$	1.00
	Random	$-68.934 \pm 0.135 \blacktriangledown$	$1.11 \times 10^3 \pm 7.48 \times 10^2 \blacktriangledown$	$684.997 \pm 506.298 \blacktriangledown$	$143.092 \pm 348.344 \blacktriangledown$	4.88
	Reg. Evolution	$-68.935 \pm 0.156 \blacktriangledown$	$2.98 \times 10^3 \pm 1.55 \times 10^3 \blacktriangledown$	$1.88 \times 10^3 \pm 1.11 \times 10^3 \blacktriangledown$	$1.38 \times 10^3 \pm 1.42 \times 10^3 \blacktriangledown$	2.50
	Simple Mutation	$-68.925 \pm 0.160 \blacktriangledown$	$1.99 \times 10^3 \pm 1.20 \times 10^3 \blacktriangledown$	$1.34 \times 10^3 \pm 8.12 \times 10^2 \blacktriangledown$	$565.382 \pm 902.668 \blacktriangledown$	3.88
	GA	-68.908 ± 0.143	$2.59 \times 10^3 \pm 1.45 \times 10^3 \blacktriangledown$	$1.80 \times 10^3 \pm 1.08 \times 10^3 \blacktriangledown$	$1.08 \times 10^3 \pm 1.29 \times 10^3 \blacktriangledown$	2.75

One potential concern is that EEM-NAS may benefit from an asymmetry in mutation proposal handling relative to the mutation baselines. Specifically, EEM-NAS forces at least one mutated edge when the entropy-guided Bernoulli mutation mask selects no edges, whereas the original simple mutation and GA operators may occasionally generate no-edit mutation proposals. To control for this effect, simple mutation and GA are additionally evaluated using the same at-least-one-edit fallback mechanism while all the other search operators remain unchanged. When no edges are selected for mutation, a single edge is forcibly mutated, with both the edge and the replacement operation chosen uniformly at random. The fairness-corrected results are summarized in Table 4. Substituting the forced-edit variants yields 81/15/0 overall, including 18/6/0 against simple mutation and 17/7/0 against GA. Regularized evolution changes only because Holm correction is recomputed for this family. The 15 ties comprise eight SWAP, six Jacov, and one NWOT cells; their mean differences are small relative to run variability, and no cell is a loss. These

results suggest that the observed advantage is not explained solely by EEM-NAS avoiding no-edit mutation proposals.

Table 4. Fairness-corrected result after substituting forced-edit simple mutation and GA into the 96-cell family; the other baselines and decision rule are unchanged.

Baseline Policy	Win	Tie	Loss
Random	24	0	0
Reg. evolution	22	2	0
Simple mutation + forced edit	18	6	0
GA + forced edit	17	7	0
Total	81	15	0

4.4. Component Analysis

Figure 3 presents the primary component analysis across three stages. Local search with reinsertion records 72/24/0, adding rank-plus-distance targeting yields 74/22/0, and the full configuration with entropy-guided mutation yields 93/3/0 against the same baselines under the same FFC budget. The increasingly favorable outcome counts support the framework design across these stages; because the stages are not compared directly, they do not isolate independent component effects.

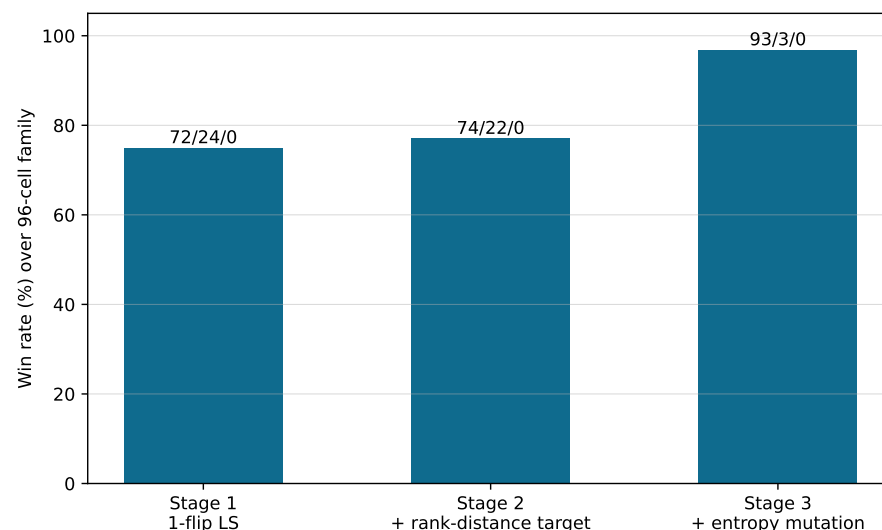


Figure 3. Proxy-score outcomes across the three stages over 96 baseline comparisons. Labels show win/tie/loss counts; all stages optimize one fixed proxy per run.

As a complementary diagnostic, Table 5 reports a leave-one-out analysis using the same initialization, FFC budget, and $n = 100$ paired seeds. The four ablations $\times$ 24 dataset-proxy cells form a separate Holm-corrected family of 96 comparisons. Removing the local-search phase, together with its target-selection step, and reallocating the freed budget to offspring evaluation degrades the final proxy score in 18 of the 24 cells (median Cliff's $\delta = +0.424$); the six non-significant cells occur on Jacov and SWAP. The other ablations produce no Holm-significant change. Thus, the local-search phase has the largest detected effect on final proxy score, while entropy-guided mutation is supported primarily by the diversity diagnostics below. These ablation effects are not additive.

To examine the diversity effect of entropy-guided mutation, Table 6 compares structural diversity at FFC = 100. The uniform mutation variant keeps the same at-least-one-edit fallback but samples the mutated edge and replacement operation uniformly. The proxy-only LS targeting variant removes the rank-plus-distance term and selects local-search

targets using proxy rank alone. We measure both unique queried architectures and the retained population multiset ($N = 10$), separating what the search inspects from what it keeps. Because a graph clustering coefficient requires an arbitrary adjacency threshold, we instead report the interquartile range (IQR) of pairwise Hamming distances as a threshold-free dispersion statistic. EEM-NAS has the highest mean pairwise distance, centroid distance, and edge entropy on both bases, whereas the IQR differences are small and overlap in standard error on the queried basis. These results are descriptive diagnostics rather than guarantees on the entropy of the accepted population.

Table 5. Leave-one-out component analysis at FFC = 100 over 24 dataset-proxy cells ($n = 100$ paired seeds). Win means that full EEM-NAS is higher with $p_{\text{Holm}} < 0.05$ and $\delta > 0$ within this 96-comparison family. The local-search ablation also inactivates its target-selection step; global-worst replacement is applied to both offspring insertion and local-search write-back.

Removed Component	Win	Tie	Loss	Median δ [Q1, Q3]
Local-search phase	18	6	0	+0.424 [+0.266, +0.508]
Entropy-guided mutation	0	24	0	−0.022 [−0.066, +0.001]
Rank-plus-distance targeting	0	24	0	+0.009 [−0.010, +0.029]
Restricted replacement window	0	24	0	−0.030 [−0.048, +0.043]

Table 6. Structural diversity at FFC = 100 (mean $\pm$ one standard error over 24 dataset-proxy cell means; $n = 100$ seeds per cell). IQR is a dispersion statistic and is not ordered as “higher is better.”

Variant	Pairwise d_H (Mean)	Pairwise d_H (IQR)	Centroid Distance	Edge Entropy
Queried architectures				
EEM-NAS	3.697 ± 0.014	1.875 ± 0.020	3.655 ± 0.014	0.750 ± 0.002
Uniform mutation	3.613 ± 0.011	1.852 ± 0.018	3.571 ± 0.011	0.733 ± 0.002
Proxy-only LS targeting	3.654 ± 0.012	1.883 ± 0.021	3.611 ± 0.012	0.744 ± 0.002
Retained population				
EEM-NAS	2.507 ± 0.041	1.506 ± 0.046	2.256 ± 0.037	0.398 ± 0.006
Uniform mutation	2.240 ± 0.038	1.471 ± 0.042	2.016 ± 0.034	0.350 ± 0.006
Proxy-only LS targeting	2.468 ± 0.035	1.489 ± 0.039	2.221 ± 0.032	0.392 ± 0.005

In addition to the final diversity statistics, Figure 4 tracks the four indicators over the queried architectures throughout the search process. EEM-NAS and uniform mutation share the same initialization phase, so both start from the same values at FFC = 10. After reproduction and local refinement begin, EEM-NAS maintains higher edge entropy over the queried architectures than uniform mutation throughout the search horizon. At FFC = 50, the macro-average entropy is 0.798 for EEM-NAS and 0.772 for uniform mutation. By FFC = 100, the corresponding values remain 0.750 and 0.733, respectively. This trajectory supports the interpretation that entropy-guided mutation maintains broader diversity among the queried architectures during low-budget search.

Figure 5 shows the same indicators for the retained population. At FFC = 100, EEM-NAS exceeds uniform mutation by 0.267 ± 0.015 in mean pairwise distance and 0.047 ± 0.003 in edge entropy, confirming that the queried-set result is also present in the maintained population.

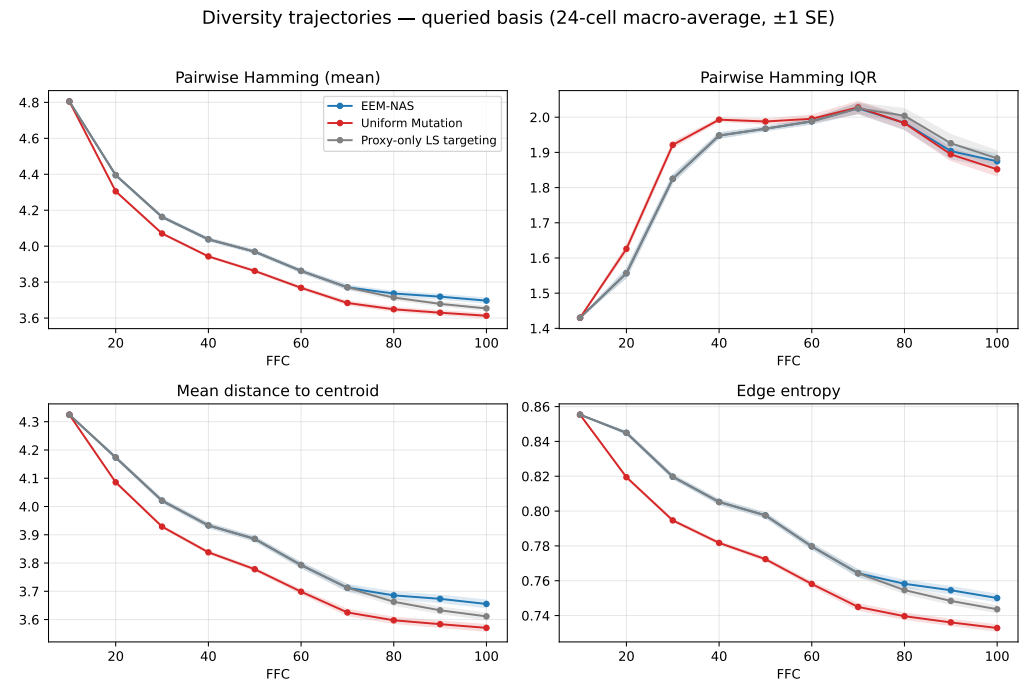


Figure 4. Diversity of unique queried architectures over FFC for three variants, macro-averaged over 24 dataset-proxy cells ($n = 100$); bands show ± 1 standard error.

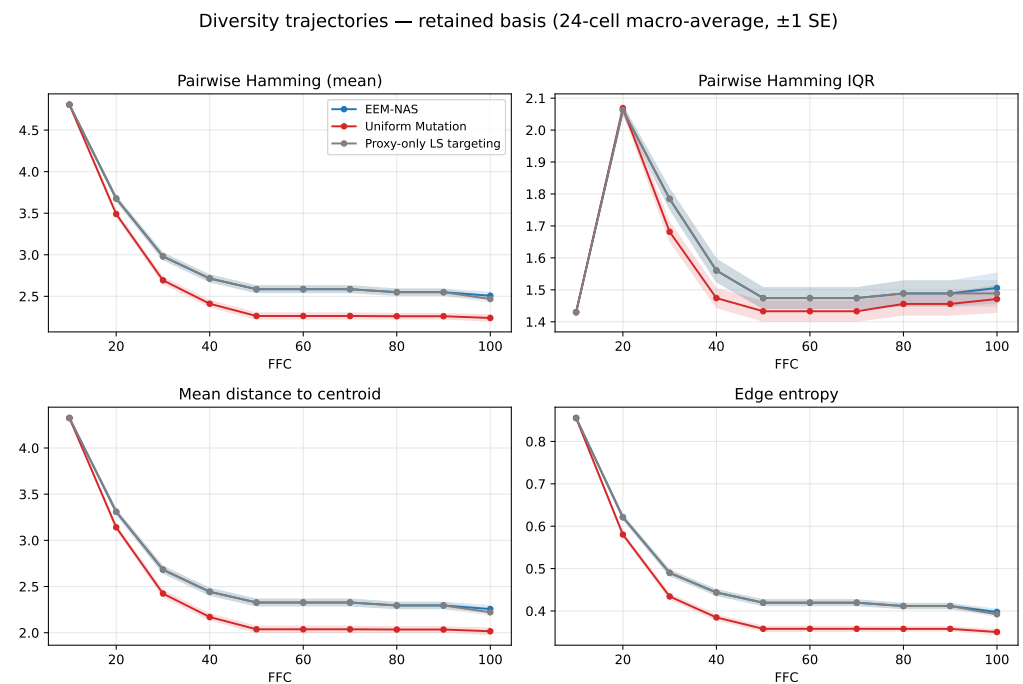


Figure 5. Retained-population diversity over FFC for three variants, macro-averaged over 24 dataset-proxy cells ($N = 10$ and $n = 100$); bands show ± 1 standard error.

4.5. Proxy Alignment and Accuracy Diagnostic

Figure 6 shows the proxy-test rank alignment of the eight evaluated zero-cost proxies on NAS-Bench-201 TSS. ZiCo, SWAP, and NWOT have the three highest mean Spearman correlations and are therefore used for the test-accuracy diagnostic.

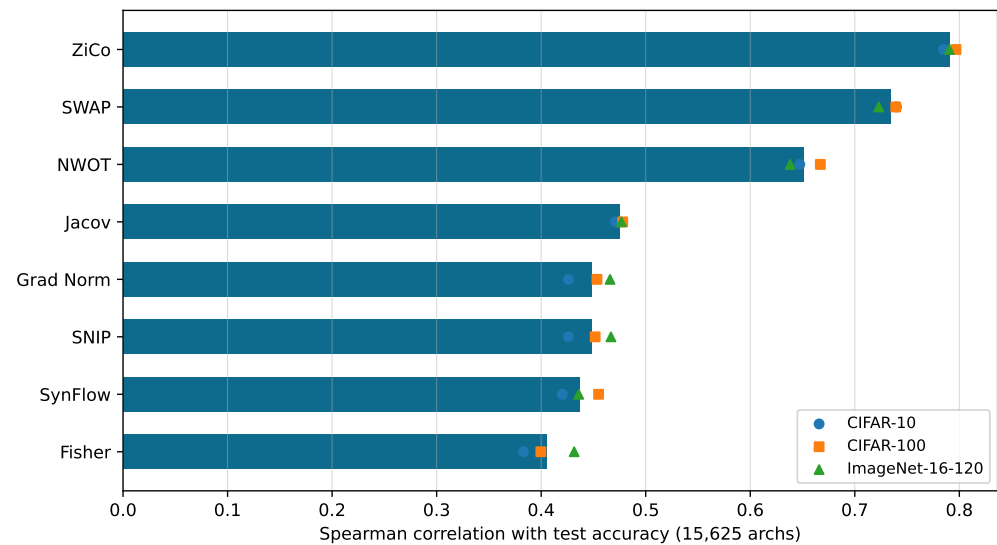


Figure 6. Proxy–test Spearman correlation over all 15,625 TSS architectures. Bars show three-dataset means and markers show individual datasets.

We descriptively split the eight proxies at the median dataset-mean Spearman correlation ($\rho = 0.4619$): ZiCo, SWAP, NWOT, and Jacov form the higher-alignment stratum, and the remaining proxies form the lower stratum. The split is not a proxy-quality threshold. Table 7 reports the resulting alignment-stratified re-aggregation.

Table 7. Alignment-stratified re-aggregation at FFC = 100. W/T/L decisions and effect sizes are inherited from the original 96-cell family; no tests are recomputed within strata.

Alignment Stratum	Proxy–Dataset Conditions	Comparisons	W/T/L	Median δ [Q1, Q3]
Higher (mean $\rho \geq 0.4619$)	12	48	45/3/0	+0.415 [+0.241, +0.614]
Lower (mean $\rho < 0.4619$)	12	48	48/0/0	+0.693 [+0.623, +0.869]
Total	24	96	93/3/0	–

The higher- and lower-alignment strata yield 45/3/0 and 48/0/0, respectively, with positive median effects. This descriptive split shows that proxy-score gains are not confined to higher alignment, but it does not test between-stratum differences. To complement this alignment-based view, Table 8 descriptively re-aggregates the same comparison family by proxy signal category.

Table 8. Descriptive re-aggregation by proxy signal at FFC = 100. W/T/L decisions and effects are inherited from the 96-cell family; no within-category tests are recomputed. The one-proxy Jacobian category precludes broad family generalization.

Signal Category	Proxies	Proxy–Dataset Conditions	Comparisons	W/T/L	Median δ
Gradient-related	5	15	60	60/0/0	+0.693
Activation-based	2	6	24	23/1/0	+0.376
Jacobian-based	1	3	12	10/2/0	+0.242
Total	8	24	96	93/3/0	–

The gradient-related, activation-based, and Jacobian-based categories yield 60/0/0, 23/1/0, and 10/2/0, respectively. Because Jacov is the only Jacobian proxy, these descriptive values do not establish family-level differences. These analyses concern fixed-proxy

optimization. The separate accuracy diagnostic uses ZiCo, SWAP, and NWOT solely because they have the three highest mean correlations.

Figure 7 presents the proxy-ranked top-10 accuracy diagnostic on ZiCo, SWAP, and NWOT. For each run, the unique queried architectures are ranked by the same proxy optimized during that run, and the test accuracies of the top-10 architectures are averaged. This diagnostic examines whether the high-proxy region discovered during search also contains architectures with high measured test accuracy. EEM-NAS has the highest observed top-10 accuracy in six of nine conditions and the highest nine-condition macro-average. Its largest exception is SWAP on ImageNet-16-120, where it trails the best mean by 0.40 percentage points.

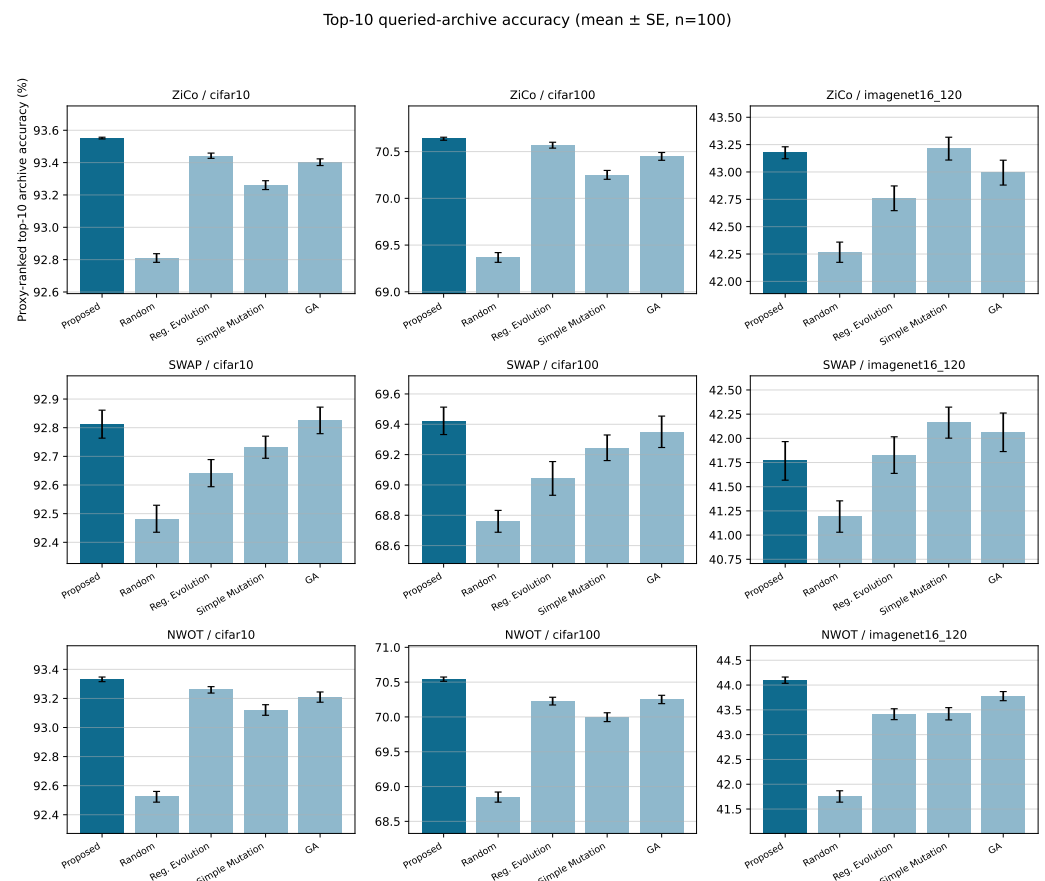


Figure 7. Proxy-ranked top-10 average test accuracy among the architectures queried at $\text{FFC} = 100$ under ZiCo, SWAP, and NWOT. Each bar averages the per-run top-10 diagnostic over $n = 100$ paired random seeds, with standard-error bars. The top 10 are selected using the same proxy optimized by the corresponding run.

We extend the accuracy diagnostic to all eight proxies using the same $n = 100$ paired seeds. The eight proxies $\times$ three datasets $\times$ four baselines = 96 comparisons form a separate Holm-corrected family. Table 9 reports 23/25/0 in the higher-alignment stratum and 5/27/16 in the lower-alignment stratum despite the 48/0/0 lower-stratum outcome on final proxy score. Thus, improved fixed-proxy optimization does not reliably translate into higher downstream accuracy when proxy–test alignment is weak. The stratum-level summaries are descriptive, and no between-stratum inferential test is performed.

Table 9. Alignment-stratified proxy-ranked top-10 accuracy at FFC = 100. The 96 cell-level comparisons form a separate Holm-corrected paired family. Win/loss denotes a Holm-significant positive/negative EEM-NAS effect; all remaining cells are ties. Stratum-level W/T/L and Cliff's δ summaries are descriptive.

Alignment Stratum	Proxy-Dataset Conditions	Comparisons	W/T/L	Median δ [Q1, Q3]
Higher (mean $\rho \geq 0.4619$)	12	48	23/25/0	+0.264 [+0.026, +0.475]
Lower (mean $\rho < 0.4619$)	12	48	5/27/16	−0.248 [−0.344, −0.078]
Total	24	96	28/52/16	—

4.6. Budget Sensitivity Analysis

To examine whether the FFC = 100 operating point remains stable across nearby budgets, we sweep the per-run FFC budget from 50 to 150 in steps of 10 on ZiCo, SWAP, and NWOT, the three most rank-aligned proxies identified in Section 4.5. The sweep focuses on the low-budget range around the FFC = 100 main comparison, where the difference between search procedures remains visible. At larger budgets, the inter-method margins become smaller as more architectures can be queried. Figure 8 shows the running-best ZiCo score over $n = 100$ paired random seeds. At FFC = 100, EEM-NAS is ahead of the strongest baseline mean by approximately 23–27 ZiCo points across the three datasets. EEM-NAS achieves the highest mean ZiCo score in 32 of the 33 evaluated dataset–budget settings, the only exception being CIFAR-100 at FFC = 50, where regularized evolution is higher by 7.0 ZiCo points.

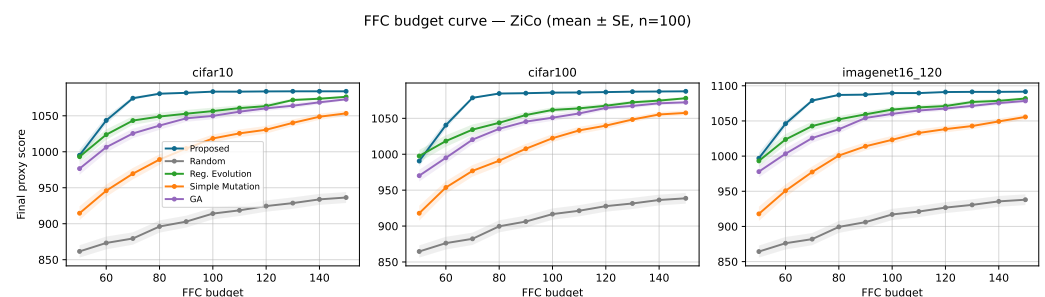


Figure 8. Running-best ZiCo proxy score versus FFC budget on the three NAS-Bench-201 datasets. Each curve indicates the mean over $n = 100$ paired random seeds for FFC values from 50 to 150 in steps of 10, with shaded bands at ± 1 standard error of the mean. Color, line style, and marker shape are varied jointly per method so that overlapping segments remain distinguishable. This sweep is a budget-sensitivity diagnostic and is separate from the 96-cell Holm-corrected FFC = 100 main comparison family.

Figure 9 reports the budget sweep on SWAP. At FFC = 100, EEM-NAS is ahead of the strongest baseline mean by approximately 1.1×10^3 to 5.9×10^3 SWAP-score units across the three datasets. EEM-NAS achieves the highest mean SWAP score across all 33 evaluated dataset–budget settings.

Figure 10 extends the budget sweep to NWOT. At FFC = 100, EEM-NAS is ahead of the strongest baseline mean by approximately 0.36–0.58 NWOT-score units across the three datasets. EEM-NAS achieves the highest mean NWOT score across all 33 evaluated dataset–budget settings. The NWOT margins are numerically smaller than those of ZiCo because the NWOT score scale is more compressed. Nevertheless, the same trend persists across the sweep, indicating that the observed budget sensitivity is not specific to ZiCo.

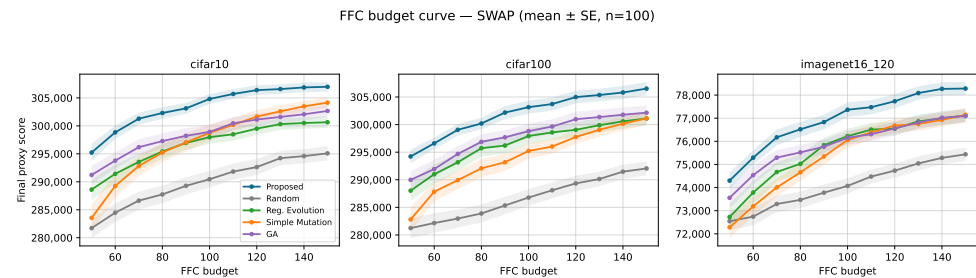


Figure 9. Running-best SWAP proxy score versus FFC budget on the three NAS-Bench-201 datasets. Each curve shows the mean over $n = 100$ paired random seeds for FFC values from 50 to 150 in steps of 10, with shaded bands at ± 1 standard error of the mean. The same sweep protocol as Figure 8 is used.

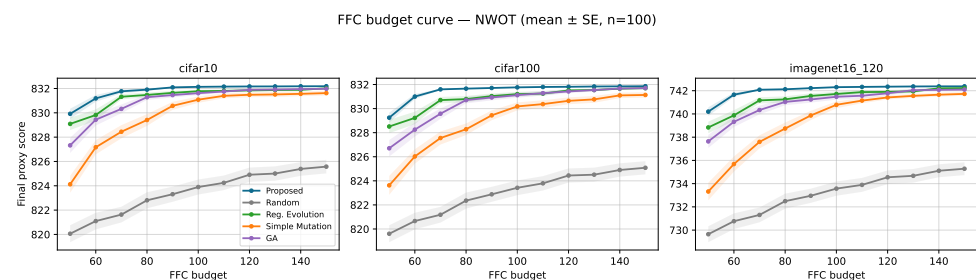


Figure 10. Running-best NWOT proxy score versus FFC budget on the three NAS-Bench-201 datasets. Each curve shows the mean over $n = 100$ paired random seeds for FFC values from 50 to 150 in steps of 10, with shaded bands at ± 1 standard error of the mean. Color, line style, and marker shape are varied jointly per method so that overlapping segments remain distinguishable. The same sweep protocol as Figure 8 is used.

4.7. Hyperparameter Sensitivity

The choice of the local-search period K_{gen} is important under $\text{FFC} = 100$ because the local-search phase competes directly with reproduction for the same proxy-call budget. To examine this sensitivity, we vary $K_{\text{gen}} \in \{1, 2, 3, 4, 5, 6, 8\}$ while keeping all the other EEM-NAS settings fixed. Each point averages $n = 30$ paired random seeds over the three datasets for each proxy. The seed count is reduced from the $n = 100$ used in the main comparison to make the seven-point evaluation across all the proxies and datasets computationally manageable. This analysis is diagnostic and is not included in the 96-cell Holm-corrected main comparison family.

Figure 11 indicates that $K_{\text{gen}} = 4$ provides the best overall balance between evolutionary search and local refinement in this low-budget setting. Specifically, it receives 15 of the 24 cell-level best-value votes and yields the best dataset-averaged mean on five of the eight proxies, namely ZiCo, NWOT, SNIP, Grad Norm, and Fisher. Earlier local-search triggers spend a larger fraction of the budget on exploitation before the population has accumulated sufficiently promising structure. Later triggers leave less budget for local refinement after the population has been shaped by reproduction. In particular, $K_{\text{gen}} = 8$ receives no cell-level vote and reduces the ZiCo mean from 1086.52 at $K_{\text{gen}} = 4$ to 1055.75. SWAP attains its highest dataset-averaged mean at $K_{\text{gen}} = 1$, although the differences across the swept values remain within one pooled standard error for this proxy. Overall, this analysis supports $K_{\text{gen}} = 4$ as the final operating point under the fixed FFC budget.

Figure 12 reports the sensitivity to the local-search budget b_{LS} . Because a no-improvement full-neighborhood scan costs 24 new evaluations, we add $b_{\text{LS}} = 24$ to the sweep grid. Values below 24 generally reduce final proxy score. Among $b_{\text{LS}} \in \{24, 25, 30\}$, a separate Holm family of 72 paired tests detect no significant difference in any cell; median Cliff's δ values are -0.002 , -0.033 , and -0.029 for 24 versus 25, 24 versus 30, and 25 versus 30, respectively. The adopted value 25 is therefore retained as an operating point

that permits one post-improvement evaluation beyond a complete scan, not as a uniquely optimal setting.

K_{gen} sensitivity (FFC=100, $n=30$ seeds, datasets pooled per proxy)

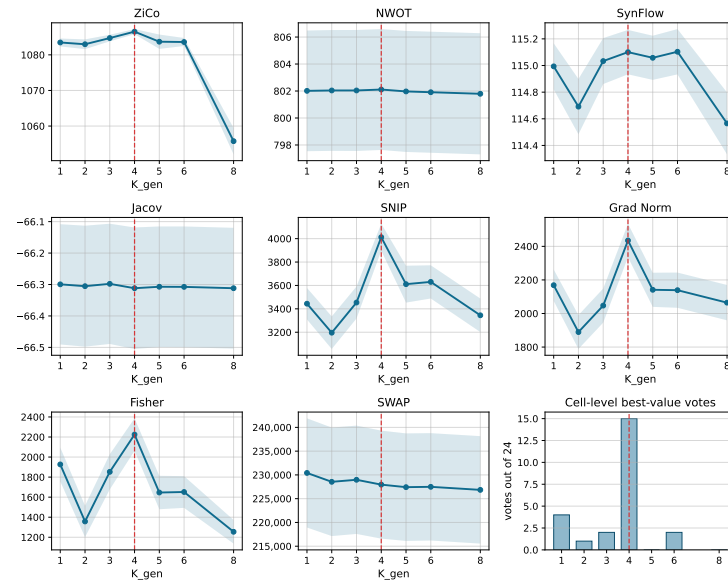


Figure 11. K_{gen} sensitivity of EEM-NAS at FFC = 100. Each proxy panel reports the final proxy score averaged over the three NAS-Bench-201 datasets and $n = 30$ paired random seeds per dataset. Shaded bands denote ± 1 standard error over the 90 runs per point. The dashed red line marks the final setting $K_{\text{gen}} = 4$. The final panel reports the scale-invariant cell-level best-value vote over the 24 dataset-proxy cells.

b_{LS} sensitivity with $b_{\text{LS}} = 24$ added (FFC=100, $n=30$ seeds, datasets pooled per proxy)

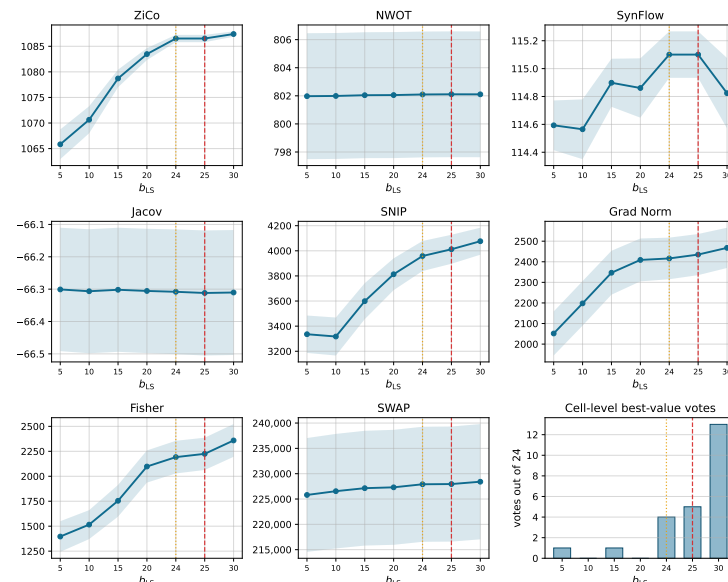


Figure 12. b_{LS} sensitivity at FFC = 100, averaged over three datasets and $n = 30$ seeds per dataset. Swept values are equally spaced horizontally for legibility; bands show ± 1 standard error. The dashed and dotted lines mark $b_{\text{LS}} = 25$ and 24 , respectively, and the last panel reports best-value votes over 24 cells.

4.8. Comparison with Bayesian Optimization

We compare EEM-NAS with Optuna's default Tree-Structured Parzen Estimator (TPE) [39,40], treating the six edges as categorical variables. Both methods optimize one proxy for 100 FFCs on ZiCo, SWAP, and NWOT over three datasets and $n = 100$ seeds; random sampling is included, and the 18 comparisons form a separate Holm family. Surrogate-fitting time is not charged. This standard TPE baseline is not a comprehensive comparison with predictor-based BO-NAS methods such as BANANAS [41]. The resulting comparison is summarized in Table 10.

Table 10. TPE comparison at FFC = 100 on three proxies (mean $\pm$ std; $n = 100$). Bold marks the best mean; the last two columns give Cliff's δ . All 18 comparisons are Holm-corrected wins.

Dataset	Proxy	EEM-NAS		TPE		Random	δ vs. TPE	δ vs. Random
CIFAR-10	ZiCo	$1.08 \times 10^3 \pm 6.05 \times 10^0$	$1.00 \times 10^3 \pm 7.03 \times 10^1$			914.086 ± 80.371	+0.848	+0.996
CIFAR-10	SWAP	$3.05 \times 10^5 \pm 8.72 \times 10^3$	$2.99 \times 10^5 \pm 1.08 \times 10^4$			$2.90 \times 10^5 \pm 1.48 \times 10^4$	+0.342	+0.575
CIFAR-10	NWOT	832.135 ± 0.238	830.051 ± 4.096			823.898 ± 6.162	+0.663	+0.941
CIFAR-100	ZiCo	$1.09 \times 10^3 \pm 7.41 \times 10^0$	$1.02 \times 10^3 \pm 6.46 \times 10^1$			916.650 ± 79.194	+0.790	+0.995
CIFAR-100	SWAP	$3.03 \times 10^5 \pm 1.19 \times 10^4$	$2.96 \times 10^5 \pm 1.44 \times 10^4$			$2.87 \times 10^5 \pm 1.51 \times 10^4$	+0.305	+0.609
CIFAR-100	NWOT	831.763 ± 0.279	828.770 ± 4.699			823.422 ± 6.139	+0.618	+0.957
ImageNet-16-120	ZiCo	$1.09 \times 10^3 \pm 8.66 \times 10^0$	$1.02 \times 10^3 \pm 6.28 \times 10^1$			917.020 ± 80.451	+0.811	+0.992
ImageNet-16-120	SWAP	$7.74 \times 10^4 \pm 3.15 \times 10^3$	$7.55 \times 10^4 \pm 2.81 \times 10^3$			$7.41 \times 10^4 \pm 3.35 \times 10^3$	+0.327	+0.507
ImageNet-16-120	NWOT	742.304 ± 0.342	739.702 ± 4.222			733.572 ± 6.131	+0.652	+0.939

EEM-NAS records 18/0/0 against TPE and random sampling, with median Cliff's $\delta = 0.726$ (range 0.305–0.996). TPE improves on random sampling but remains below EEM-NAS in every cell.

4.9. Cross-Space Diagnostic on NATS-Bench SSS

The main experiments are based on the NAS-Bench-201 TSS encoding. To examine whether the EEM-NAS search process remains competitive on a different tabulated search space, we additionally evaluate EEM-NAS and the four baselines on the size search space (SSS) of NATS-Bench [42]. In NATS-Bench SSS, an architecture is defined by the channel widths of five sequential layers, where each layer selects one width from eight candidate values. This yields $8^5 = 32,768$ architectures. The corresponding one-flip neighborhood contains $5 \times (8 - 1) = 35$ neighbors, which is larger than the 24-neighbor Hamming-1 neighborhood of NAS-Bench-201 TSS but remains comparable in scale. With the local-search cap $b_{LS} = 25$, one local-search call can inspect most of a one-flip neighborhood in this space. This supports using the same local-search cap for the cross-space diagnostic. This experiment is intended as a single-dataset diagnostic rather than a second full benchmark. We therefore vary the search space while holding the dataset fixed to CIFAR-10. We evaluate ZiCo, SWAP, and NWOT with online proxy computation at FFC = 100 and $n = 100$ paired seeds. The 12 EEM-NAS-to-baseline comparisons form a separate Holm family.

Table 11 shows that EEM-NAS remains competitive on NATS-Bench SSS. EEM-NAS has the highest mean for all three proxies (12/0/0 against the baselines) and the highest observed top-10 accuracy mean in all three conditions. Consistent with the NAS-Bench-201 TSS results, this diagnostic suggests that the EEM-NAS budget-allocation strategy remains competitive on a second tabulated space with comparable one-flip neighborhood scale.

Table 11. NATS-Bench SSS diagnostic on CIFAR-10 with online proxies at FFC = 100 (mean $\pm$ std). Bold values indicate the highest mean within each proxy and metric.

Proxy	Method	Final Proxy Score	Top-10 Acc. (%)
ZiCo	EEM-NAS	302.436 $\pm$ 2.523	93.01 $\pm$ 0.09
	Random	294.551 $\pm$ 3.252	92.57 $\pm$ 0.13
	Reg. Evolution	301.536 $\pm$ 2.993	92.94 $\pm$ 0.10
	Simple Mutation	299.519 $\pm$ 3.111	92.88 $\pm$ 0.10
	GA	300.609 $\pm$ 2.868	92.94 $\pm$ 0.10
SWAP	EEM-NAS	2.94 $\times 10^5 \pm 2.07 \times 10^3$	93.07 $\pm$ 0.12
	Random	2.84 $\times 10^5 \pm 5.39 \times 10^3$	92.10 $\pm$ 0.24
	Reg. Evolution	2.93 $\times 10^5 \pm 4.20 \times 10^3$	92.99 $\pm$ 0.13
	Simple Mutation	2.89 $\times 10^5 \pm 4.95 \times 10^3$	92.81 $\pm$ 0.23
	GA	2.92 $\times 10^5 \pm 3.44 \times 10^3$	92.94 $\pm$ 0.17
NWOT	EEM-NAS	791.180 $\pm$ 0.317	93.19 $\pm$ 0.10
	Random	788.302 $\pm$ 1.474	92.19 $\pm$ 0.21
	Reg. Evolution	790.791 $\pm$ 0.988	93.08 $\pm$ 0.13
	Simple Mutation	789.868 $\pm$ 1.462	92.87 $\pm$ 0.20
	GA	790.778 $\pm$ 0.853	93.02 $\pm$ 0.16

5. Discussion

The main results evaluate fixed-proxy search performance. EEM-NAS allocates a limited proxy-evaluation budget more effectively than the compared fixed-proxy baselines, and the 93/3/0 win/tie/loss result directly measures this objective. This result should be distinguished from a direct claim about final architecture accuracy. A higher proxy score is useful for architecture discovery only when the optimized proxy is sufficiently aligned with trained test accuracy.

Let P_k and A_k be the proxy- and accuracy-ranked top- k sets after deterministic tie breaking, and let $D = \sum_i (r_i - s_i)^2$. A mismatch of q top- k items requires $D \geq 2q^3$, giving

$$|P_k \cap A_k| \geq \max\left\{0, 2k - n, k - \left\lfloor (D/2)^{1/3} \right\rfloor\right\}.$$

Using $D = n(n^2 - 1)(1 - \rho)/6$, this lower bound is zero for $n = 15,625$, $k \in \{10, 50, 100\}$, and all 24 conditions; Spearman correlation alone therefore gives no nonzero top- k guarantee here. The resulting normalized proxy-accuracy top- k overlaps are reported in Table 12.

Table 12. Normalized proxy-accuracy top- k overlap over 15,625 architectures. Ties use a stable descending sort; mean ρ is repeated for reference.

Proxy	Mean ρ	CIFAR-10			CIFAR-100			ImageNet-16-120		
		$k = 10$	$k = 50$	$k = 100$	$k = 10$	$k = 50$	$k = 100$	$k = 10$	$k = 50$	$k = 100$
ZiCo	0.791	0.00	0.00	0.04	0.00	0.02	0.10	0.00	0.12	0.17
SWAP	0.734	0.00	0.02	0.09	0.00	0.04	0.05	0.00	0.00	0.04
NWOT	0.651	0.00	0.00	0.02	0.00	0.02	0.07	0.00	0.12	0.22
Jacov	0.475	0.00	0.04	0.05	0.00	0.04	0.06	0.10	0.04	0.06
Grad Norm	0.448	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04
SNIP	0.448	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04
SynFlow	0.437	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.02
Fisher	0.405	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03

The measured overlap is also low: at $k = 10$, 23 of the 24 conditions are disjoint; the only nonzero value is 0.10 for Jacov on ImageNet-16-120. Overlap increases mainly for better-aligned proxies at $k = 50$ and 100, reaching 0.22 for NWOT on ImageNet-16-120.

Thus, proxy optimization guides search toward a high-quality region but cannot guarantee recovery of the accuracy-ranked top set.

To examine the practical consequence of this limitation, test accuracy is reported separately as a search-level proxy-ranked top-10 diagnostic on ZiCo, SWAP, and NWOT, the three most rank-aligned proxies in the evaluated set. It averages test accuracy over the ten highest-proxy architectures queried by each method. It does not imply that stronger fixed-proxy optimization resolves proxy–test misalignment for all proxies. Thus, the practical interpretation is that EEM-NAS improves the search of a fixed proxy, while the usefulness of the discovered architectures still depends on the reliability of that proxy. The accuracy-level stratification in Table 9 quantifies this dependence. The top-10 accuracy advantage is concentrated in the higher-alignment stratum (23/25/0), whereas the lower-alignment stratum yields 5/27/16. This supports reading the main result strictly as a claim about budgeted fixed-proxy search rather than about downstream accuracy.

The component analysis further clarifies the scope of the method’s contribution. EEM-NAS is not a new zero-cost proxy, local-search operator, crossover operator, or multi-proxy fusion method. Instead, it is a memetic search framework that coordinates bounded local refinement, rank-plus-distance target selection, RTS-based reinsertion, and entropy-guided mutation under a shared proxy-evaluation budget. Figure 3 presents the results across the three stages, while Table 5 complements them with marginal ablations on the same paired seeds. Removing the local-search phase produces Holm-significant degradation in 18 of the 24 cells, whereas the other ablations show no detected marginal effect on final proxy score. The diversity diagnostics over the queried architectures and the retained population support the interpretation that entropy-guided mutation broadens the set of architectures visited and maintained under the same proxy-call budget. These results support the framework-level contribution of combining non-redundant refinement, selective reinsertion, and population-level exploration rather than attributing the full gain to a single primitive.

The strict-better replacement rule yields only weak finite-budget monotonicity. If M_q is the maximum retained proxy score after q evaluations, accepted replacements cannot remove the current maximum, and rejected proposals do not change the population; hence

$$M_{q+1} \geq M_q \quad (N \leq q < B_{\text{FFC}}), \quad M_{B_{\text{FFC}}} - M_N \geq 0.$$

This does not guarantee strict improvement or a unique fixed point: under a constant proxy, every initial population is absorbing. Stronger claims require assumptions on the proxy landscape and proposal probabilities that are not made here.

The observed behavior is also tied to the geometry of the search space. The main experiments use the NAS-Bench-201 TSS encoding, where a Hamming-1 neighborhood contains 24 neighbors. This neighborhood size is closely matched to the local-search cap $b_{\text{LS}} = 25$, allowing one local-search call to inspect a substantial portion of the immediate neighborhood. The cross-space diagnostic on NATS-Bench SSS provides a limited check of this design choice on a second tabulated space with a comparable one-flip neighborhood size. In that diagnostic, EEM-NAS attains the highest mean on all three proxies and separates clearly from random sampling. This suggests that the budget-allocation strategy is not restricted to the NAS-Bench-201 TSS encoding alone. NATS-Bench SSS is only a limited diagnostic on a 2.1-times-larger tabulated space with a comparable neighborhood. Much larger spaces may weaken coverage and require retuning b_{LS} , K_{LS} , and K_{gen} or sampling the one-flip neighborhood; performance there remains unverified. Different encodings, learned predictors, hardware-aware objectives, and multi-objective search may also require different search biases. The current evidence should therefore be read as a low-budget fixed-proxy

search result centered on NAS-Bench-201 TSS with a supporting cross-space diagnostic on NATS-Bench SSS rather than as a general statement about all memetic NAS settings.

6. Conclusions

In this study, we formulated zero-shot NAS on NAS-Bench-201 TSS as a budgeted fixed-proxy search problem and introduced Evaluation-Efficient Memetic NAS (EEM-NAS). EEM-NAS coordinates bounded one-flip local refinement, rank-plus-distance target selection, RTS-based reinsertion, and entropy-guided mutation under a shared proxy-evaluation budget. The method is designed to balance targeted exploitation of promising architectures with population-level exploration while retaining useful local refinements through sampled-window replacement. At $\text{FFC} = 100$, EEM-NAS achieved 93 statistically significant wins, three ties, and zero losses over the 96-cell Holm-corrected comparison family on final proxy score. A mutation-controlled fairness diagnostic yielded a more conservative 81/15/0 result. The component analysis progresses from 72/24/0 to 93/3/0, while the complementary leave-one-out analysis identifies the local-search phase as the largest detected marginal contributor to final proxy score. On ZiCo, SWAP, and NWOT, the three most rank-aligned proxies, EEM-NAS achieved the highest macro-average proxy-ranked top-10 accuracy among the compared methods. Across all eight proxies, the corresponding accuracy comparison yields 23/25/0 in the higher-alignment stratum and 5/27/16 in the lower-alignment stratum. EEM-NAS also outperformed the standard TPE in all nine TPE comparisons within the 18-comparison Holm-corrected family. The b_{LS} sweep, extended with the exact full-sweep value $b_{\text{LS}} = 24$, detected no Holm-significant difference among $b_{\text{LS}} \in \{24, 25, 30\}$ on final proxy score, so the adopted value is an operating point rather than a uniquely optimal setting.

These results support EEM-NAS as an effective search procedure for low-budget fixed-proxy zero-shot NAS. At the same time, the accuracy of the discovered architectures remains proxy-dependent because stronger proxy optimization does not by itself resolve proxy–test misalignment. Future work should evaluate the framework on larger and non-tabulated NAS spaces, additional evaluation budgets, and online proxy-computation settings. Another important direction is to study proxy reliability jointly with fixed-proxy search allocation, especially when the proxy signal is weakly aligned with final architecture accuracy.

Author Contributions: Conceptualization, S.K. and W.S.; methodology, S.K. and W.S.; software, S.K.; validation, S.K.; formal analysis, S.K.; investigation, S.K.; resources, W.S.; data curation, S.K.; writing—original draft preparation, S.K.; writing—review and editing, W.S.; visualization, S.K.; supervision, W.S.; project administration, W.S.; funding acquisition, W.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Kyonggi University Research Grant 2025.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: NAS-Bench-201 TSS is publicly available from the original benchmark release [34]. A reference implementation of EEM-NAS, the four budgeted fixed-proxy search baselines, the eight zero-cost proxy backends, and the experiment scripts that auto-download the NAS-Bench-201 release file are available at <https://github.com/minercode625/eem-nas> (accessed on 29 July 2026).

Acknowledgments: The authors used AI-assisted writing tools to polish the academic style and language of the manuscript.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

BO	Bayesian Optimization
FFC	Fitness Function Call
GA	Genetic Algorithm
LS	Local Search
EEM-NAS	Evaluation-Efficient Memetic NAS
NAS	Neural Architecture Search
RTS	Restricted Tournament Selection
TSS	Topology Search Space
NWOT	Neural Architecture Search Without Training
SWAP	Sample-Wise Activation Patterns
TPE	Tree-Structured Parzen Estimator
ZiCo	Zero-Cost Proxy Based on Gradient Coefficient Statistics

References

1. Salmani Pour Avval, S.; Eskue, N.D.; Groves, R.M.; Yaghoubi, V. Systematic review on neural architecture search. *Artif. Intell. Rev.* **2025**, *58*, 73. <https://doi.org/10.1007/s10462-024-11058-w>.
2. Poyser, M.; Breckon, T.P. Neural architecture search: A contemporary literature review for computer vision applications. *Pattern Recognit.* **2024**, *147*, 110052. <https://doi.org/10.1016/j.patcog.2023.110052>.
3. Chaiyarin, S.; Rojbundit, N.; Piyabenjarad, P.; Limpitgranon, P.; Wisitthipakdeekul, S.; Nonthasaen, P.; Achararit, P. Neural architecture search for medicine: A survey. *Inform. Med. Unlocked* **2024**, *50*, 101565.
4. Xing, Z.; Li, L.; Yang, Y.; Zhou, W.; Ma, G.; Chen, S.; Lyu, L.; Li, X. A lightweight model for indoor object detection in unstructured scenes based on joint attention and prior knowledge in the context of home rehabilitation. *J. King Saud Univ. Comput. Inf. Sci.* **2026**, *38*, 186. <https://doi.org/10.1007/s44443-026-00605-w>.
5. Xing, Z.; Meng, Z.; Zheng, G.; Yang, L.; Guo, X.; Tan, L.; Jiang, Y. Human-computer interactive rehabilitation: A 3D graph deep learning method for non-contact gesture recognition in post-epidemic and aging societies. *Measurement* **2026**, *257*, 118794. <https://doi.org/10.1016/j.measurement.2025.118794>.
6. Elsken, T.; Metzen, J.H.; Hutter, F. Neural Architecture Search: A Survey. *J. Mach. Learn. Res.* **2019**, *20*, 1–21.
7. Mellor, J.; Turner, J.; Storkey, A.; Crowley, E.J. Neural Architecture Search without Training. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*; PMLR: Cambridge, MA, USA, 2021; Volume 139, pp. 7588–7598.
8. Abdelfattah, M.S.; Mehrotra, A.; Dudziak, L.; Lane, N.D. Zero-Cost Proxies for Lightweight NAS. In *Proceedings of the International Conference on Learning Representations (ICLR)*, Virtual Event, Austria, 3–7 May 2021.
9. Li, G.; Yang, Y.; Bhardwaj, K.; Marculescu, R. ZiCo: Zero-shot NAS via Inverse Coefficient of Variation on Gradients. In *Proceedings of the International Conference on Learning Representations (ICLR)*, Kigali, Rwanda, 1–5 May 2023.
10. He, X.; Zhao, K.; Chu, X. AutoML: A Survey of the State-of-the-Art. *Knowl.-Based Syst.* **2021**, *212*, 106622.
11. Real, E.; Aggarwal, A.; Huang, Y.; Le, Q.V. Regularized Evolution for Image Classifier Architecture Search. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*; AAAI: Washington, DC, USA, 2019; Volume 33, pp. 4780–4789.
12. Moscato, P. *On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms*; Technical Report C3P Report 826, Caltech Concurrent Computation Program; California Institute of Technology: Pasadena, CA, USA, 1989.
13. Neri, F.; Cotta, C. Memetic algorithms and memetic computing optimization: A literature review. *Swarm Evol. Comput.* **2012**, *2*, 1–14. <https://doi.org/10.1016/j.swevo.2011.11.003>.
14. Dong, J.; Hou, B.; Feng, L.; Tang, H.; Tan, K.C.; Ong, Y.S. A Cell-Based Fast Memetic Algorithm for Automated Convolutional Neural Architecture Design. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *34*, 9040–9053. <https://doi.org/10.1109/TNNLS.2022.3155230>.
15. Qiao, Y.; Xu, H.; Zhang, Y.; Huang, S. MONAS: Efficient zero-shot neural architecture search for MCUs. In *Proceedings of the 2025 International Joint Conference on Neural Networks (IJCNN)*; IEEE: New York, NY, USA, 2025; pp. 1–8.
16. Harik, G.R. Finding Multimodal Solutions Using Restricted Tournament Selection. In *Proceedings of the 6th International Conference on Genetic Algorithms (ICGA)*; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1995; pp. 24–31.
17. Tanaka, H.; Kunin, D.; Yamins, D.L.K.; Ganguli, S. Pruning Neural Networks without Any Data by Iteratively Conserving Synaptic Flow. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 6377–6389.
18. Lee, N.; Ajanthan, T.; Torr, P.H.S. SNIP: Single-Shot Network Pruning Based on Connection Sensitivity. In *Proceedings of the International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA, 6–9 May 2019.

19. Theis, L.; Korshunova, I.; Tejani, A.; Huszár, F. Faster Gaze Prediction with Dense Networks and Fisher Pruning. *arXiv* **2018**, arXiv:1801.05787.
20. Turner, J.; Crowley, E.J.; O’Boyle, M.; Storkey, A.; Gray, G. BlockSwap: Fisher-Guided Block Substitution for Network Compression on a Budget. In Proceedings of the International Conference on Learning Representations (ICLR), Addis Ababa, Ethiopia, 26–30 April 2020.
21. Peng, Y.; Song, A.; Fayek, H.M.; Ciesielski, V.; Chang, X. SWAP-NAS: Sample-Wise Activation Patterns for Ultra-fast NAS. In Proceedings of the International Conference on Learning Representations (ICLR), Vienna, Austria, 7–11 May 2024.
22. Jiang, T.; Wang, H.; Bie, R. MeCo: Zero-Shot NAS with One Data and Single Forward Pass via Minimum Eigenvalue of Correlation. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 61020–61047.
23. Lee, J.; Ham, B. AZ-NAS: Assembling Zero-Cost Proxies for Network Architecture Search. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 17–21 June 2024.
24. Cortês, G.; Lourenço, N.; Romano, P.; Machado, P. GreenFactory: Ensembling Zero-Cost Proxies to Estimate Performance of Neural Networks. *arXiv* **2025**, arXiv:2505.09344.
25. Wu, N.; Huang, H.; Xu, Y.; Hao, Z. Zero-Shot NAS via the Suppression of Local Entropy Decrease. *arXiv* **2024**, arXiv:2411.06236.
26. Whitley, D.; Gordon, V.S.; Mathias, K. Lamarckian evolution, the Baldwin effect and function optimization. In *Proceedings of the Parallel Problem Solving from Nature—PPSN III*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1994; Volume 866, pp. 5–15. https://doi.org/10.1007/3-540-58484-6_245.
27. Mei, J.; Gulyás, L.; Botzheim, J. Comparing Lamarckian and Baldwinian Approaches in Memetic Optimization. In *Proceedings of the Advances in Computational Collective Intelligence*; Communications in Computer and Information Science, Springer: Cham, Switzerland, 2023; Volume 1864, pp. 521–533. https://doi.org/10.1007/978-3-031-41774-0_41.
28. Bereta, M. Baldwin effect and Lamarckian evolution in a memetic algorithm for Euclidean Steiner tree problem. *Memetic Comput.* **2019**, *11*, 35–52. <https://doi.org/10.1007/s12293-018-0256-7>.
29. Santana, C.; Oliveira, M.; Bastos-Filho, C.; Menezes, R. Beyond exploitation: Measuring the impact of local search in swarm-based memetic algorithms through the interactions of individuals in the population. *Swarm Evol. Comput.* **2022**, *70*, 101040. <https://doi.org/10.1016/j.swevo.2022.101040>.
30. Subburaj, B.; Amali, S.M.J. A fuzzy system based self-adaptive memetic algorithm using population diversity control for evolutionary multi-objective optimization. *Sci. Rep.* **2025**, *15*, 5735. <https://doi.org/10.1038/s41598-025-89289-2>.
31. Cotta, C. Harnessing memetic algorithms: A practical guide. *TOP* **2025**, *33*, 327–356. <https://doi.org/10.1007/s11750-024-00694-8>.
32. Phan, Q.M.; Luong, N.H. Enhancing Multi-objective Evolutionary Neural Architecture Search with Training-free Pareto Local Search. *Appl. Intell.* **2023**, *53*, 8654–8672. <https://doi.org/10.1007/s10489-022-04032-y>.
33. Phan, Q.M.; Luong, N.H. Efficient Multi-Fidelity Neural Architecture Search with Zero-Cost Proxy-Guided Local Search. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*; ACM: New York, NY, USA, 2024.
34. Dong, X.; Yang, Y. NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search. In Proceedings of the International Conference on Learning Representations (ICLR), Addis Ababa, Ethiopia, 26–30 April 2020.
35. Wilcoxon, F. Individual Comparisons by Ranking Methods. *Biom. Bull.* **1945**, *1*, 80–83. <https://doi.org/10.2307/3001968>.
36. Demšar, J. Statistical Comparisons of Classifiers over Multiple Data Sets. *J. Mach. Learn. Res.* **2006**, *7*, 1–30.
37. Holm, S. A Simple Sequentially Rejective Multiple Test Procedure. *Scand. J. Stat.* **1979**, *6*, 65–70.
38. Romano, J.; Kromrey, J.D.; Coraggio, J.; Skowronek, J. Appropriate Statistics for Ordinal Level Data: Should We Really Be Using *t*-Test and Cohen’s *d* for Evaluating Group Differences on the NSSE and Other Surveys? In Proceedings of the Annual Meeting of the Florida Association of Institutional Research, Cocoa Beach, FL, USA, 1–3 February 2006; pp. 1–33.
39. Bergstra, J.; Bardenet, R.; Bengio, Y.; Kégl, B. Algorithms for Hyper-Parameter Optimization. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*; Curran Associates Inc.: Red Hook, NY, USA, 2011; Volume 24.
40. Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; Koyama, M. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*; ACM: New York, NY, USA, 2019; pp. 2623–2631.
41. White, C.; Neiswanger, W.; Savani, Y. BANANAS: Bayesian Optimization with Neural Architectures for Neural Architecture Search. In *Proceedings of the AAAI Conference on Artificial Intelligence*; AAAI: Washington, DC, USA, 2021; Volume 35, pp. 10293–10301.
42. Dong, X.; Liu, L.; Musial, K.; Gabrys, B. NATS-Bench: Benchmarking NAS Algorithms for Architecture Topology and Size. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 3634–3646. <https://doi.org/10.1109/TPAMI.2021.3054824>.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.