Bi-EZP: LLM-Guided Bilevel Program Evolution for Ensemble Zero-Cost Proxy Discovery

Yutao Lai, Kezhao Lai, Hai-Lin Liu

***Abstract*—Zero-cost proxies enable neural architecture search (NAS) to rank candidate networks from statistics computed at initialization, avoiding the repeated training required by conventional performance estimation. Their efficiency, however, comes with a reliability limitation: different proxies measure different properties of an untrained network and often induce inconsistent rankings across search spaces. Ensemble proxies can combine complementary signals, but automated ensemble discovery must determine both a discrete aggregation structure and the continuous coefficients associated with that structure. Searching these variables jointly makes structural quality difficult to distinguish from parameter calibration. We propose Bi-EZP, a bilevel framework that separates the two decisions. At the upper level, a large language model generates executable aggregation programs over four complementary base proxies together with program-specific parameter bounds. At the lower level, covariance matrix adaptation evolution strategy (CMA-ES) optimizes the continuous parameters of each fixed program on an inner training split. The calibrated programs are compared using Kendall's rank correlation on a disjoint validation split, and evolutionary selection retains structures that generalize beyond their calibration data. The discovery phase is supervised by benchmark architecture accuracies, whereas the resulting frozen proxy evaluates new architectures without training them to convergence. Experiments on NATS-Bench and Network Design Spaces assess rank correlation across heterogeneous search spaces, while experiments in the DARTS space examine its use in downstream architecture search. The reported results show that explicitly separating program discovery from numerical calibration provides an effective route to automated ensemble zero-cost proxy construction within the evaluated settings. The source code is available at: https://anonymous.4open.science/r/Bi-EZP-318D.**

***Index Terms*—Neural architecture search, zero-shot NAS, zero-cost proxy, ensemble proxy, bilevel optimization, large language model, CMA-ES.**

I. Introduction

Neural architecture search (NAS) replaces hand-designed network construction with an optimization process over architectural choices [1]. Reinforcement-learning methods [2], [3], evolutionary search [4], [5], and differentiable optimization [6]–[8] have all produced competitive architectures. Their progress has also exposed a persistent bottleneck: architecture evaluation dominates the search cost. Training every candidate is prohibitively expensive, while weight sharing reduces but does not remove the computational burden and can distort the relative quality of candidates.

Zero-shot NAS addresses this bottleneck by estimating architecture quality without full candidate training [9]. Zero-cost proxies extract statistics from a randomly initialized network using a small number of forward or backward passes. Parameter saliency [10], [11], activation patterns [12], feature correlations [13], and gradient consistency [14] provide inexpensive signals that can be reused by different NAS search algorithms. Yet these signals encode distinct inductive biases. Large-scale evaluations show that a proxy that is informative in one search space may be weak or even misleading in another [15], [16]. The central difficulty is therefore not only to design another scalar heuristic, but to combine heterogeneous proxy evidence without committing to a brittle aggregation rule.

Existing ensemble and automated proxy methods provide two complementary routes. Fixed-form ensembles combine a prescribed set of signals through voting, ranking, or parameterized aggregation [17], [18]. Symbolic approaches instead search over expressions or executable programs [19]–[21]. The former are numerically tractable but structurally restricted; the latter enlarge the structural space but introduce structure-dependent continuous parameters. This creates a coupled optimization problem. A useful program may appear weak when evaluated with poorly chosen coefficients, whereas extensive tuning can make a restricted structure appear competitive. Consequently, comparing uncalibrated structures does not isolate the value of the aggregation rule itself.

Recent advances in large language models (LLMs) have demonstrated capabilities beyond natural-language generation, particularly in optimization-related tasks that require structured reasoning and executable representations. LLMs have been investigated for generating optimization code [22], constructing black-box optimization benchmarks [23], and directly producing candidate solutions for complex optimization problems [24]. These studies suggest that LLMs can serve as flexible generators of structured optimization artifacts rather than merely text generators. This capability is particularly relevant to zero-shot proxy discovery, where the search object is not only a set of numerical coefficients but also a symbolic program specifying nonlinear transformations and interactions among multiple proxy signals. Compared with a manually predefined grammar, an LLM provides a more flexible structural prior that can generate diverse yet executable aggregation expressions.

Our key observation is that symbolic structure discovery and continuous parameter fitting play fundamentally different roles and should therefore be optimized at different levels. The symbolic structure determines which transformations and interactions among base proxies are expressible, whereas its parameter vector determines how a fixed structure behaves on a particular discovery sample. Bi-EZP formalizes this

Yutao Lai, Kezhao Lai, and Hai-Lin Liu are with the School of Mathematics and Statistics, Guangdong University of Technology, Guangzhou 510006, China (e-mail: 2112114007@mail2.gdut.edu.cn; 3124006080@mail2.gdut.edu.cn; hlliu@gdut.edu.cn)

distinction as a bilevel optimization problem. The upper level searches executable aggregation programs together with their feasible parameter domains. For every proposed program, the lower level independently calibrates its numerical parameters before the program receives an upper-level fitness. This nested evaluation makes the comparison between candidate structures conditional on dedicated parameter optimization rather than on arbitrary initial coefficients.

Within this framework, Bi-EZP employs an LLM specifically as a structural generator rather than as a numerical optimizer or an autonomous NAS decision maker. Guided by prompts that encourage nonlinear transformations and cross-proxy interactions, the LLM proposes and varies candidate aggregation programs. A parser then constrains each response to a fixed four-proxy interface and an explicit set of finite parameter bounds, ensuring syntactic and numerical validity. Given a generated structure, CMA-ES optimizes its continuous parameters within these bounds, while validation Kendall's τ provides the upper-level fitness for tournament selection and elitist replacement. In this way, the LLM supplies flexible structural diversity, whereas CMA-ES and rank-based evolutionary selection provide explicit performance-driven optimization and validation.

The problem also differs from direct LLM-based architecture generation [25]–[28]. Direct generation proposes one architecture at a time, whereas proxy discovery learns a reusable evaluation function that can score many candidates and can be embedded in different NAS procedures. The discovery phase uses benchmark accuracies as supervision; it is therefore not training-free in its entirety. Once the selected program and its parameters are frozen, however, evaluating another candidate requires only its initialization-time proxy statistics rather than full network training.

The main contributions of this work are summarized as follows:

- We formulate automated ensemble zero-cost proxy discovery as a bilevel optimization problem that explicitly decouples symbolic aggregation structure discovery from structure-dependent continuous parameter calibration. The upper level evolves executable proxy programs, while the lower level uses CMA-ES to optimize the continuous parameters of each fixed structure, enabling candidate structures to be compared after dedicated numerical calibration rather than under arbitrary coefficient settings.
- We propose a parameterized prompting mechanism that enables LLMs to generate executable symbolic proxy structures together with their corresponding parameter constraints for lower-level numerical optimization. Each generated candidate jointly specifies an aggregation program and program-specific parameter bounds, which are validated through an executable-code checking and correction mechanism before being passed to the CMA-ES optimizer, thereby connecting open-ended LLM-based program generation with bounded numerical optimization.
- We conduct extensive experiments on NATS-Bench, NDS, and DARTS across CIFAR-10, CIFAR-100, and ImageNet, where Bi-EZP demonstrates competitive performance in both rank-correlation evaluation and downstream architecture search.

II. Related Work

A. Efficient Neural Architecture Search

The computational cost of candidate evaluation has shaped the development of NAS. Early reinforcement-learning and evolutionary approaches train large numbers of sampled networks [2], [4], [5]. Parameter sharing reduces this cost by evaluating subnetworks inside a common supernet [3], while differentiable methods relax discrete architectural choices into continuous variables [6]–[8]. Other approaches use predictors, partial training, or population-based optimization to allocate evaluation effort more efficiently [29]. These methods optimize architectures directly. Zero-shot NAS instead targets the evaluation function itself, replacing learned or trained performance estimates with statistics available near initialization.

B. Zero-Cost Proxies

Zero-cost proxies approximate architecture quality through inexpensive properties of an untrained network [9], [15]. Gradient-based criteria extend pruning-at-initialization signals such as SNIP and GraSP to architecture ranking [10], [11]; GradNorm and related measures summarize the magnitude or organization of initialization gradients [15]. Activation-based methods characterize the expressivity of randomly initialized mappings. NASWOT measures diversity in binary activation codes [12], TE-NAS combines neural tangent kernel conditioning with linear-region counts [30], and Zen-NAS estimates expressivity through feature perturbations [31]. More recent proxies exploit feature correlation, gradient consistency, or sample-wise activation patterns, including MeCo [13], ZiCo [14], and SWAP-NAS [32]. L-SWAG further combines activation and gradient information for vision transformers [33]. Because these criteria observe different aspects of an initialized network, their rankings need not agree. Bi-EZP takes this heterogeneity as the input to ensemble discovery rather than assuming that any single signal is sufficient.

C. Automated and Ensemble Proxy Discovery

Ensemble proxies combine complementary signals to reduce dependence on a single heuristic. AZ-NAS integrates multiple zero-cost measures into a unified score [17], while ParZC learns a parameterized representation for zero-cost prediction [34]. Per-architecture metric optimization provides another parameterized route to combining evaluation signals [35]. These approaches show that calibration matters, but their parameterizations restrict the forms of interaction that can be expressed.

Automated symbolic discovery expands the candidate space. EZNAS evolves executable proxy expressions with genetic programming [19], Auto-Prox searches training-free predictors for vision transformers [20], and symbolic regression has been used to derive performance-prediction formulas from

primitive statistics [21]. ECP is the closest setting to Bi-EZP because it combines NASWOT, MeCo, Sweet-SNIP, and SZiCo through a fixed power-function family and optimizes its coefficients with adaptive particle swarm optimization [18]. EvoREP [36] automatically evolves nonlinear ensembles of multiple zero-cost proxies to obtain more reliable architecture evaluations [36]. Bi-EZP retains this four-source basis but changes the optimization object: the aggregation program and its parameter bounds become outer-level variables, and each fixed program receives an independent inner numerical calibration. The distinction is therefore the separation of structural and parametric decisions, rather than a claim that one program-search formalism is intrinsically more compact than another.

D. LLMs for NAS and Program Search

Related work has explored several complementary ways of automating NAS with evolutionary optimization and LLMs. For example, LAPT [37] transfers design principles between search tasks or incorporate rapid proxy feedback into LLM-guided architecture search [38]. More recently, LLME-NAS [39] integrates LLM guidance into evolutionary NAS by dynamically adapting the search fitness according to historical optimization trajectories [39]. These studies demonstrate the potential of both automated proxy composition and LLM-assisted evolutionary search. Bi-EZP connects these two directions but differs fundamentally in its search object: rather than evolving a predefined combination of proxies or using the LLM to guide the search for neural architectures, it employs the LLM to generate syntactically structured architecture-scoring programs. Explicit validation, CMA-ES-based continuous calibration, and rank-based evolutionary selection then determine whether a generated scoring program remains in the population.

III. Methodology

A. Overview

Bi-EZP searches for a reusable ensemble proxy over four precomputed zero-cost signals. The method contains two coupled but separately optimized levels. The upper level explores symbolic aggregation programs, using an LLM to initialize candidates and to produce crossover or mutation variants. The lower level receives one fixed program and optimizes only its continuous parameters with CMA-ES. After calibration on inner-training architectures, the candidate is evaluated on disjoint inner-validation architectures. Its validation Kendall's τ becomes the fitness used by the outer evolutionary loop.

Figure 1 illustrates this information flow. The LLM supplies a candidate description $\mathcal{H}$, executable structure $\mathcal{G}$, and finite parameter bounds $\mathcal{B}$. Before numerical calibration, an executable-code gate checks the abstract syntax tree, required function signature, and literal `BOUNDS` list. Invalid responses undergo a limited number of correction attempts; if all attempts fail, a fixed fallback program keeps the population evaluable. Only validator-accepted pairs $(\mathcal{G}, \mathcal{B})$ define candidate-specific continuous domains $\Theta(\mathcal{G})$ for CMA-ES. The optimizer returns a calibrated vector θ^* inside that domain, after which the complete proxy $\Phi(\cdot; \mathcal{G}, \theta^*)$ is assigned a validation fitness. Thus, structural selection never compares unchecked or uncalibrated programs.

B. Base Proxy Space

The input basis is adopted from ECP [18] and is fixed as

$$\mathcal{S} = \{s_{nw}, s_{mc}, s_{ss}, s_{sz}\}, \tag{1}$$

where the four signals correspond to NASWOT, MeCo, Sweet-SNIP, and SZiCo, respectively. We select these four proxies to construct a compact yet heterogeneous input basis. They characterize an initialized architecture from complementary sources of network information: NASWOT captures activation-pattern diversity, MeCo describes feature correlation, Sweet-SNIP measures parameter saliency, and SZiCo reflects gradient consistency across input batches [18]. Together, they provide activation-, feature-, parameter-, and gradient-level views of network quality, reducing the dependence of the aggregation function on any single architectural characteristic. At the same time, these statistics can be obtained jointly using a forward pass and at most two backward propagations, which preserves the computational efficiency required by zero-shot NAS.

This paper keeps this proxy set fixed and focuses exclusively on discovering how these heterogeneous signals should be transformed and combined. The objective is therefore not to identify an optimal subset of zero-cost proxies, but to study whether automatically discovered transformations and interactions can yield a more effective architecture-scoring function than manually specified aggregation rules.

1) NASWOT (s_{nw}): NASWOT provides the activation-level component of the basis and measures network expressivity through the diversity of binary activation patterns induced by a mini-batch [12]. For ReLU layer r, let $c_i^{(r)}$ denote the binary activation code of input i, N_r the number of activation units, and d_H the Hamming distance. The layer kernel and network score are

$$K_{ij}^{(r)} = N_r - d_H\left(c_i^{(r)}, c_j^{(r)}\right), \qquad s_{nw} = \log\left|\sum_{r=1}^{R} K^{(r)}\right|. \tag{2}$$

The determinant summarizes how distinctly the initialized network partitions the sampled inputs. Unlike the remaining gradient- or feature-related signals, NASWOT depends only on activation patterns and therefore provides a complementary, gradient-free view of the architecture.

2) MeCo (s_{mc}): MeCo provides a feature-level view by quantifying redundancy in intermediate representations through the minimum eigenvalue of a Pearson correlation matrix [13]. Following the formulation used by ECP [18], four channels are sampled from the C_l channels of layer l:

$$s_{mc} = \sum_{l=1}^{L} \frac{C_l}{4} \lambda_{\min}(\mathbf{P}(\mathbf{f}_l)), \tag{3}$$

where $\mathbf{f}_l$ contains the flattened sampled features and $\mathbf{P}(\cdot)$ denotes their Pearson correlation matrix. A larger minimum eigenvalue indicates that the sampled feature directions are less degenerate. Thus, MeCo complements activation-pattern

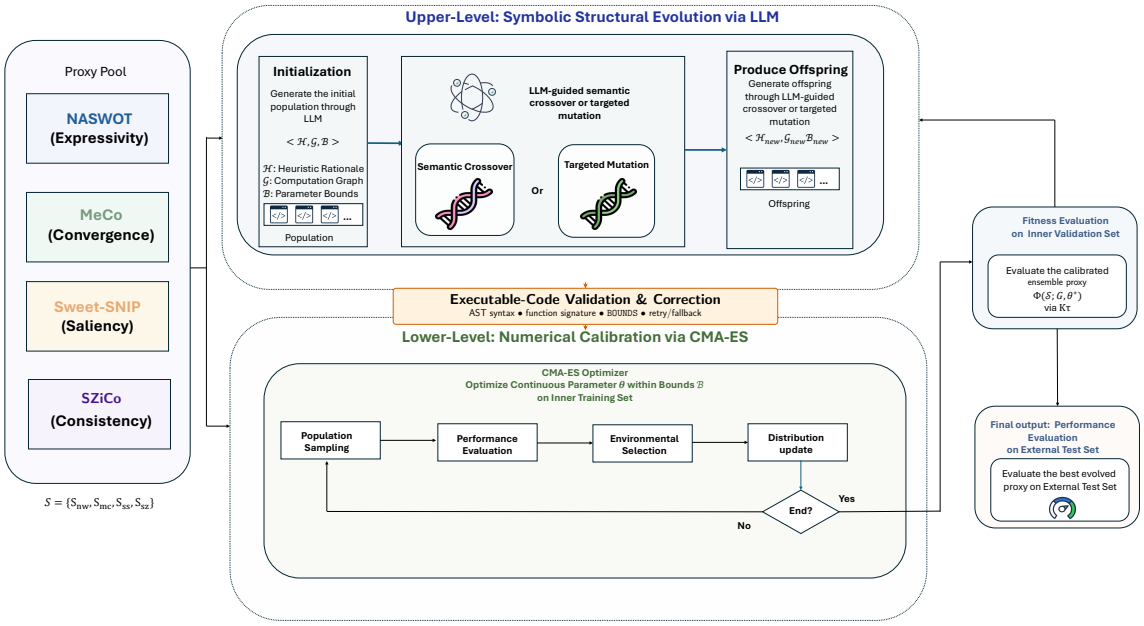

Fig. 1. Overview of Bi-EZP. The upper level proposes executable aggregation structures and parameter domains. A code-validation gate checks syntax, interface, and bounds, applying correction attempts or a fixed fallback before a candidate reaches the lower-level CMA-ES calibration. Inner-training rank correlation drives numerical calibration, and inner-validation rank correlation drives structural selection.

information by describing the geometry and redundancy of learned feature representations, which have been associated with training convergence and generalization.

3) Sweet-SNIP (s_{ss}): Sweet-SNIP supplies a parameter-saliency perspective. SNIP estimates the importance of an initialized parameter through the product between its magnitude and loss gradient [10]. ECP further incorporates the sweet-gradient mechanism [40], retaining gradients within a prescribed interval:

$$s_{ss} = \sum_{l=1}^{L} \left| \boldsymbol{\theta}_l \circ \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_l} \right| \mathbb{I}\left\{ \sigma_1 \leq \left| \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_l} \right| \leq \sigma_2 \right\}. \tag{4}$$

Here, $\circ$ denotes element-wise multiplication and $\mathbb{I}\{\cdot\}$ is an indicator function. Whereas NASWOT and MeCo characterize network responses, Sweet-SNIP directly measures the sensitivity of the loss to initialized parameters, providing information about parameter importance that is not explicitly represented by the former proxies.

4) SZiCo (s_{sz}): SZiCo provides a gradient-statistical perspective. ZiCo evaluates gradient consistency by comparing the mean gradient magnitude with its variation across input batches [14]. ECP combines this statistic with the sweet-gradient interval to obtain SZiCo [18]:

$$s_{zico} = \sum_{l=1}^{L} \log\left(\sum_{\theta \in \boldsymbol{\theta}_l} \frac{\mathbb{E}_j[|\partial_\theta \mathcal{L}(\mathbf{X}_j, \mathbf{y}_j)|]}{\sqrt{\mathrm{Var}_j(\partial_\theta \mathcal{L}(\mathbf{X}_j, \mathbf{y}_j))}} \right), \tag{5}$$

$$s_{sz} = s_{zico}\, \mathbb{I}\left\{ \sigma_1 \leq \left| \frac{\partial \mathcal{L}}{\partial \theta} \right| \leq \sigma_2 \right\}. \tag{6}$$

While Sweet-SNIP focuses on parameter-wise saliency, SZiCo measures the stability of gradient information across data batches and therefore captures a different aspect of trainability, convergence, and generalization.

C. Parameterized Proxy Representation

For architecture i, let

$$\mathbf{s}_i = [s_{nw}^{(i)}, s_{mc}^{(i)}, s_{sz}^{(i)}, s_{ss}^{(i)}]^\top \tag{7}$$

denote its base-proxy vector. A candidate program maps a matrix of such vectors to one scalar score per architecture. We represent the candidate generated by the LLM as

$$\mathcal{C} = \langle \mathcal{H}, \mathcal{G}, \mathcal{B} \rangle. \tag{8}$$

The textual component $\mathcal{H}$ records the generated rationale, $\mathcal{G}$ is executable Python code implementing the aggregation rule, and

$$\mathcal{B} = \prod_{k=1}^{d} [L_k, U_k], \qquad L_k < U_k, \tag{9}$$

defines the feasible set $\Theta(\mathcal{G})$ of the d continuous parameters used by that program. For a batch of n architectures, the executable proxy is

$$\Phi(\mathbf{S}; \mathcal{G}, \theta) : \mathbb{R}^{4\times n} \times \Theta(\mathcal{G}) \to \mathbb{R}^n. \tag{10}$$

The executable interface turns open-ended generation into an evaluable search object. Every response must contain an "Idea" section and a "Code" section. The code must define a nonempty literal list `BOUNDS` and exactly one synchronous function

```
aggregate(s_naswot, s_meco, s_zico,
          s_ssnip, params).
```

The four proxy arguments are one-dimensional arrays with a shared architecture dimension; `params` supplies the values optimized by CMA-ES. The intended return value is one finite score vector of the same length.

Before evaluation, an abstract-syntax-tree parser checks Python syntax, the exact function signature, the absence of variable or keyword-only arguments, and a literal list of finite numerical bound pairs. Rejected responses are returned to the LLM together with the validation error for a limited number of correction attempts. If all attempts fail, the implementation inserts a fixed four-proxy weighted-sum candidate so that the population remains evaluable. Runtime failures, nonfinite outputs, and undefined rank correlations receive a penalty.

The resulting structural space is operational rather than a closed symbolic grammar: it is induced jointly by the prompt, the LLM, and the validator. The current parser constrains the interface and bounds but does not impose an operator whitelist, a maximum expression depth, or an explicit complexity penalty. This distinction matters because Bi-EZP searches executable programs under interface constraints; it does not enumerate a finite set of algebraic trees.

D. Bilevel Discovery Objective

Let the supervised discovery set be

$$\mathcal{D}^{disc} = \{(\mathbf{s}_i, a_i)\}_{i=1}^{N}, \tag{11}$$

where a_i is the benchmark accuracy of architecture i. The discovery pool is partitioned into disjoint inner-training and inner-validation subsets,

$$\mathcal{D}_{train} = (\mathbf{S}_{train}, \mathbf{A}_{train}), \qquad \mathcal{D}_{val} = (\mathbf{S}_{val}, \mathbf{A}_{val}). \tag{12}$$

The implementation uses an 80/20 split inside the sampled discovery pool. Architectures outside that pool form the subsequent evaluation set.

For fixed $(\mathcal{G}, \mathcal{B})$, the lower level selects parameters that maximize rank agreement on $\mathcal{D}_{train}$:

$$\theta^*(\mathcal{G}, \mathcal{B}) \in \arg\max_{\theta \in \mathcal{B}} \tau(\Phi(\mathbf{S}_{train}; \mathcal{G}, \theta), \mathbf{A}_{train}) \,. \tag{13}$$

The upper level evaluates the calibrated program on $\mathcal{D}_{val}$ and searches over the validator-accepted candidate space $\mathbb{Q}$:

$$(\mathcal{G}^*, \mathcal{B}^*) \in \arg\max_{(\mathcal{G},\mathcal{B}) \in \mathbb{Q}} \tau\big(\Phi(\mathbf{S}_{val}; \mathcal{G}, \theta^*(\mathcal{G}, \mathcal{B})), \mathbf{A}_{val}\big). \tag{14}$$

Kendall's τ is used at both levels because the proxy is evaluated by the ordering it induces, not by calibrated prediction error. The two levels nevertheless receive different data and optimize different variables. The inner objective fits the numerical behavior of one fixed structure, whereas the outer objective compares calibrated structures by validation ranking. This separation prevents the outer loop from preferring a program merely because its initial coefficients happen to be favorable.

The bounds are part of the outer decision because different programs can require different parameter dimensionalities and numerical domains. Accordingly, a structural mutation may change the functional form, the number of parameters, their bounds, or several of these properties together. The inner solver is reinitialized for each accepted offspring; optimized parameters are not inherited across incompatible structures.

E. LLM-Guided Structural Evolution

At outer generation t, Bi-EZP maintains a population

$$\mathbb{P}^{(t)} = \{\langle \mathcal{H}_i, \mathcal{G}_i, \mathcal{B}_i, \theta_i^*, f_i \rangle\}_{i=1}^{P}, \tag{15}$$

where f_i is the validation Kendall correlation of the calibrated program. Initialization requests P independent candidates from a common prompt. Each valid candidate is calibrated on $\mathcal{D}_{train}$ before its first fitness is computed on $\mathcal{D}_{val}$, so the initial population follows the same nested evaluation protocol as later offspring.

Parent selection uses tournaments of size three among candidates whose fitness is valid. With probability P_c, two selected parents are inserted into a crossover prompt that requests a new program combining or modifying their mathematical components. Otherwise, one parent is inserted into a mutation prompt that requests a targeted structural change, such as a different nonlinear transformation or interaction. Both operators must return the same executable interface and a compatible `BOUNDS` list.

The numerical fitness does not appear directly in the crossover or mutation prompt. It influences generation through tournament selection: better validated structures are more likely to supply the code shown to the LLM. Likewise, the stored rationale $\mathcal{H}$ is retained for logging but is not supplied as persistent semantic memory to later prompts. The LLM therefore acts as a prompt-conditioned program proposal and variation operator. Structural survival is determined by external execution, CMA-ES calibration, and validation correlation.

Every generation produces P offspring. After validation and lower-level calibration, parents and offspring are merged, sorted by f_i, and truncated to the best P candidates. This elitist $(P + P)$ replacement retains the strongest programs found so far while maintaining continued exploration through newly generated code. The process terminates after a fixed number of outer cycles, and the highest-fitness calibrated candidate is returned.

F. Lower-Level Parameter Adaptation via CMA-ES

Once the LLM formulates the mathematical tuple $\langle \mathcal{H}, \mathcal{G}, \mathcal{B} \rangle$, the hypothesis space is strictly bounded. Identifying the optimal continuous coefficients within this specific parameter manifold $\Theta(\mathcal{G})$ is subsequently delegated to the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [41]. We use CMA-ES because its derivative-free covariance adaptation is compatible with the rugged, non-differentiable, and potentially ill-conditioned rank-correlation objective.

The lower-level optimization is meticulously initialized based on the explicit priors $\mathcal{B}$ mapped by the LLM. For an n-dimensional parameter space bounded by the Cartesian product $\mathcal{B} = \prod_{i=1}^{n} [L_i, U_i]$, the initial search distribution mean $m^{(0)}$ is geometrically centered within the bounds, and the

initial global step size $\sigma^{(0)}$ is scaled proportionally to the average boundary range:

$$m^{(0)} = \frac{L+U}{2}, \quad \sigma^{(0)} = \max\left(\frac{1}{4n}\sum_{i=1}^{n}(U_i - L_i), 0.01\right) \tag{16}$$

At generation g, CMA-ES samples a population of λ parameter vectors from a multivariate normal distribution parameterized by the current mean $m^{(g)}$ and covariance matrix $C^{(g)}$:

$$\theta_k^{(g+1)} \sim m^{(g)} + \sigma^{(g)}\mathcal{N}\left(0, C^{(g)}\right) \quad \text{for } k = 1, \ldots, \lambda \tag{17}$$

To rigorously evaluate each sampled parameter vector θ_k, the framework computes the composite proxy scores $\Phi(\mathcal{S}_{train}; \mathcal{G}, \theta_k)$ strictly on the inner-training set $\mathcal{D}_{train}$. Because LLM-generated symbolic expressions may occasionally produce undefined mathematical operations, we introduce a strict penalty mechanism. The objective function minimized by the lower-level solver is defined as:

$$\mathcal{L}(\theta_k) = \begin{cases} -\tau\left(\Phi(\mathcal{S}_{train}; \mathcal{G}, \theta_k), \mathcal{A}_{train}\right), & \text{if } \Phi \text{ is valid} \\ M, & \text{if } \Phi \text{ yields NaN/Inf} \end{cases} \tag{18}$$

where M is a large penalty scalar that immediately forces the evolutionary trajectory away from mathematically unstable parameter regions. Based on the μ best-performing parameter vectors, CMA-ES updates the covariance matrix $C^{(g+1)}$ using both rank-μ and rank-1 updates. The complete algorithmic implementation is formalized in Algorithm 2.

Algorithm 1 The Proposed Framework of Bi-EZP

Input: $\mathcal{D}_{train}$, $\mathcal{D}_{val}$, population size P, outer cycles T, crossover probability P_c
Output: $\langle \mathcal{H}^*, \mathcal{G}^*, \mathcal{B}^*, \theta^* \rangle$

$\mathbb{P}^{(0)} \leftarrow$ LLM-INITIALIZE(P)
for each $\langle \mathcal{H}_i, \mathcal{G}_i, \mathcal{B}_i \rangle \in \mathbb{P}^{(0)}$ **do**
  $(\mathcal{G}_i, \mathcal{B}_i) \leftarrow$ VALIDATE-OR-CORRECT($\mathcal{G}_i, \mathcal{B}_i$)
  $\theta_i^* \leftarrow$ CMA-ES($\mathcal{G}_i, \mathcal{B}_i, \mathcal{D}_{train}$)
  $f_i \leftarrow \tau(\Phi(\mathbf{S}_{val}; \mathcal{G}_i, \theta_i^*), \mathbf{A}_{val})$
end for
for $t = 0$ to $T - 1$ **do**
  $\mathbb{O} \leftarrow \emptyset$
  for $j = 1$ to P **do**
    $p_1 \leftarrow$ TOURNAMENTSELECT($\mathbb{P}^{(t)}$)
    if rand() $< P_c$ **then**
      $p_2 \leftarrow$ TOURNAMENTSELECT($\mathbb{P}^{(t)}$)
      $c \leftarrow$ LLM-CROSSOVER($\mathcal{G}_{p_1}, \mathcal{G}_{p_2}$)
    else
      $c \leftarrow$ LLM-MUTATION($\mathcal{G}_{p_1}$)
    end if
    $c \leftarrow$ VALIDATE-OR-CORRECT(c)
    $\theta_c^* \leftarrow$ CMA-ES($\mathcal{G}_c, \mathcal{B}_c, \mathcal{D}_{train}$)
    $f_c \leftarrow \tau(\Phi(\mathbf{S}_{val}; \mathcal{G}_c, \theta_c^*), \mathbf{A}_{val})$
    $\mathbb{O} \leftarrow \mathbb{O} \cup \{\langle c, \theta_c^*, f_c \rangle\}$
  end for
  $\mathbb{P}^{(t+1)} \leftarrow$ TOPP($\mathbb{P}^{(t)} \cup \mathbb{O}$)
end for
return the highest-fitness member of $\mathbb{P}^{(T)}$

G. Nested Search Procedure

Algorithm 1 summarizes the interaction between the two optimization levels. The essential ordering is calibration before structural comparison. A newly generated program is first checked for syntactic and interface validity. CMA-ES then optimizes the parameters defined by its own bounds using only $\mathcal{D}_{train}$. The resulting parameter vector is frozen while the program receives its outer fitness on $\mathcal{D}_{val}$. Only this validation fitness participates in tournament selection and elitist replacement.

This ordering prevents validation information from entering the numerical objective of CMA-ES. It also ensures that a program is never carried into the population with an unevaluated parameter vector. If code generation fails repeatedly, the fallback candidate passes through the same lower- and upper-level evaluation path; it is not assigned a privileged fitness. Algorithm 2 gives the lower-level procedure used for each accepted candidate.

H. Discovery Protocol and Frozen-Proxy Evaluation

Bi-EZP distinguishes proxy *discovery* from subsequent proxy *use*. Discovery is supervised: benchmark accuracies in $\mathbf{A}_{train}$ guide CMA-ES, and benchmark accuracies in $\mathbf{A}_{val}$ guide structural selection. Describing the complete discovery process as training-free would therefore be inaccurate. The zero-cost property applies after discovery, when $\mathcal{G}^*$ and θ^* have been fixed and the resulting proxy scores a candidate network from initialization-time statistics.

The evaluation protocol first samples a discovery pool from a benchmark search space, then partitions that pool into the inner-training and inner-validation subsets used by the two optimization levels. Architectures outside the discovery pool are reserved for final evaluation. For an architecture x in this held-out set, evaluation computes the four base proxies and applies

$$\hat{s}(x) = \Phi(\mathbf{s}(x); \mathcal{G}^*, \theta^*). \tag{19}$$

Neither the aggregation structure nor its parameters are updated during this stage. Cross-space or cross-dataset evaluation follows the same requirement: the complete proxy must be frozen before scores from the target setting are observed.

The distinction also clarifies the role of the DARTS experiments. Bi-EZP does not replace the architecture-search algorithm. It provides the search procedure with an evaluation signal that ranks candidate architectures without training each candidate to convergence. The cost of discovering this signal is incurred once, whereas the frozen aggregation program can subsequently score many architectures wherever the required base proxies are available.

I. Computational Characteristics and Scope

For population size P and T outer cycles, the procedure evaluates $P(T+1)$ candidate programs when every cycle produces a full offspring population. With at most r generation

Algorithm 2 Lower-Level Parameter Calibration via CMA-ES

Input: Decoupled Sub-tuple $\langle \mathcal{G}, \mathcal{B} \rangle$ where $\mathcal{B} = \prod_{i=1}^{n}[L_i, U_i]$, Inner-training data $\mathcal{D}_{train} = \{\mathcal{S}_{train}, \mathcal{A}_{train}\}$, Max evaluations E_{max}

Output: Optimal continuous parameters θ^*

Initialize:
Extract bounds L, U from Cartesian product $\mathcal{B}$
$m \leftarrow (L + U)/2$
$\sigma \leftarrow \max\left(\frac{1}{4n}\sum_i (U_i - L_i), 0.01\right)$
$C \leftarrow \mathbf{I}_{n \times n}$ *% Initial covariance matrix*
$e \leftarrow 0$ *% Evaluation counter*
while $e < E_{max}$ and not converged **do**
  Sample λ offspring: $\theta_k \sim m + \sigma\mathcal{N}(0, C)$ for $k = 1, \ldots, \lambda$
  for $k = 1$ to λ **do**
    $\theta_k \leftarrow \text{Clip}(\theta_k, L, U)$ *% Enforce explicit LLM boundaries $\mathcal{B}$*
    Compute proxy scores $\mathbf{s}_k = \Phi(\mathcal{S}_{train}; \mathcal{G}, \theta_k)$
    if $\mathbf{s}_k$ contains NaN or Inf **then**
      $\mathcal{L}(\theta_k) \leftarrow 999.0$ *% Penalty for instability*
    else
      $\tau_k \leftarrow \text{KendallTau}(\mathbf{s}_k, \mathcal{A}_{train})$
      $\mathcal{L}(\theta_k) \leftarrow 1 - \tau_k$
    end if
    $e \leftarrow e + 1$
  end for
  Sort the population such that $\mathcal{L}(\theta_{1:\lambda}) \leq \mathcal{L}(\theta_{2:\lambda}) \cdots \leq \mathcal{L}(\theta_{\lambda:\lambda})$
  Update mean m using the top μ vectors: $m \leftarrow \sum_{i=1}^{\mu} w_i \theta_{i:\lambda}$
  Update global step size σ using cumulative step-size adaptation
  Update covariance matrix C using rank-μ and rank-1 updates
end while
Return $\theta^* \leftarrow \theta_{1:\lambda}$ *% Best evaluated parameters*

or correction attempts per candidate, the number of LLM requests is bounded by $rP(T + 1)$. If CMA-ES is limited to E_{max} objective evaluations for each candidate, the inner level performs at most $E_{max}P(T + 1)$ aggregation evaluations on the discovery training split. The total discovery cost therefore combines LLM inference, base-proxy extraction for the discovery architectures, repeated aggregation evaluation inside CMA-ES, and validation evaluation of calibrated programs.

The principal modeling assumption is that the four fixed base proxies contain complementary information that an aggregation program can exploit. Bi-EZP does not establish that this pool is optimal, nor does it penalize program complexity explicitly. Moreover, the executable search space depends on the language model, prompt, decoding behavior, and validator. Reproducible use therefore requires reporting the LLM version, decoding configuration, random seeds, population and cycle counts, correction and fallback rates, CMA-ES budget, data partitions, and the final generated program with its optimized parameters.

The current implementation validates syntax, interface shape, and finite parameter bounds, but it does not enforce a closed operator grammar or reject semantically duplicate programs. These choices favor structural flexibility at the cost of a less sharply characterized search space. The empirical claims in this work are consequently restricted to the supplied four-proxy basis, the reported benchmarks, and the evaluated discovery and transfer protocols.

IV. EXPERIMENTAL STUDIES

A. Datasets and Search Space

To comprehensively evaluate the proposed framework and the discovered ensemble proxies, we conduct extensive experiments across diverse, well-established architectural search spaces and image classification datasets.

Search Spaces for Rank Correlation Evaluation. To rigorously assess the effectiveness of the generated zero-cost proxies, we measure the Kendall's rank correlation coefficient (τ) between the proxy scores and the ground-truth test accuracies. For this evaluation, we utilize two prominent benchmarks: NATS-Bench [42] and the Network Design Spaces (NDS) [43]. NATS-Bench provides a standardized environment containing both topology-based and size-based search spaces with fully trained ground-truth performances, allowing for precise, reproducible correlation analysis. NDS offers a broader set of network families (such as ResNet and ResNeXt variants), enabling us to test the structural generalization capacity of our composite proxies across highly heterogeneous architectural topologies.

DARTS Search Space. In addition to the benchmarks, we apply our automated proxy evaluation framework to the widely adopted DARTS search space. Evaluation within the continuous relaxation of DARTS tests whether the discovered proxy can serve as an architecture-ranking signal in a downstream search procedure.

Datasets. The evaluations across these search spaces are conducted on three standard visual recognition datasets: CIFAR-10, CIFAR-100 [44], and ImageNet. CIFAR-10 and CIFAR-100 serve as the primary datasets for extracting initialized network statistics, conducting the structural evolution of the proxies, and performing the lower-level numerical calibration. To further verify the robust transferability and large-scale generalization of the architectures discovered via our optimized proxies, we extend our final evaluations to the high-resolution, large-scale ImageNet dataset.

B. Implementation Details

To ensure a fair and consistent comparison, the dataset configurations and evaluation protocols strictly follow the experimental settings established in prior ensemble proxy research [18]. Specifically, to construct the fitness evaluation datasets, we randomly sample 1,000 architectures along with their corresponding accuracies on the second training set for the NATS-Bench tasks. For the NDS search spaces, a subset of 500 architectures is sampled for fitness evaluation. To construct the fitness evaluation set within the open-domain

DARTS search space, a random subset of 100 architectures and their corresponding accuracies on the CIFAR-10 second training set was sampled. Following the configuration in [18], [45], the search process on the DARTS space entails 10 search and 100 validation iterations. The discovered architectures are subsequently evaluated using the standard DARTS training pipeline [6]. During the proxy extraction phase across all tasks, the input batch size is set to 128. Furthermore, for the gradient-based base proxies (i.e., SSNIP and SZiCo), the sweet gradient intervals are uniformly set to $[1e-4, 1e-3]$ for the NATS-Bench and NDS benchmarks. For the DARTS search space, these intervals are adjusted to $[1e-5, 1e-2]$ for CIFAR-10 and $[5e-5, 1e-4]$ for ImageNet.

For the proposed bilevel optimization framework, the upper-level structural evolution, which is driven by the Large Language Model (LLM), maintains a population size of 20 individuals. Specifically, GLM-4.7-Flash [46] is employed as the underlying LLM in our experiments. The structural evolutionary search is conducted over a maximum of 20 generations. In the lower-level numerical calibration phase, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is configured according to standard CMA-ES hyperparameter settings. The continuous parameter optimization strictly adheres to the boundaries generated by the LLM, with the maximum number of fitness evaluations for CMA-ES capped at 1,500 per structural hypothesis.

The implementation initializes the Python, NumPy, and PyTorch random-number generators with seed 1 before architecture sampling and proxy discovery. CMA-ES is invoked without a separate optimizer seed, and no deterministic seed is passed to the LLM API. LLM requests use the ZhipuAI v4 chat-completions endpoint with the model identifier `glm-4.7-flash`, temperature 1.0, disabled thinking, and non-streaming output. Top-p and the maximum output-token budget are not set explicitly and therefore follow the API defaults. Generated responses are parsed and checked using the validation and correction procedure described in Section III-C.

C. Experimental Analysis

This section evaluates Bi-EZP across established neural architecture search benchmarks and open-domain search spaces. The experiments examine rank correlation, transfer across datasets and search spaces, sensitivity to design choices, and use of the proxy in downstream architecture search.

1) Experimental Result in Rank Correlation Across NAS Benchmarks: The Kendall's rank correlation coefficient (τ) serves as the primary metric for evaluating the ranking fidelity of zero-cost proxies, measuring the consistency between proxy predictions and ground-truth test accuracies. Table I presents a comprehensive comparison on the NATS-Bench benchmark, covering both topology-based (TSS) and size-based (SSS) search spaces.

Bi-EZP achieves the highest rank correlation among the compared methods under the reported protocol. Relative to ECP, Bi-EZP increases the TSS τ score from 0.782 to 0.809 on CIFAR-10, from 0.771 to 0.791 on CIFAR-100, and from 0.740 to 0.775 on ImageNet-16-120. On SSS, the corresponding correlations increase from 0.771 to 0.792 on CIFAR-10 and from 0.806 to 0.826 on ImageNet-16-120.

ECP already employs an automated symbolic-regression mechanism for proxy discovery. These results are consistent with a benefit from separating symbolic structure search and continuous parameter calibration under the reported protocol. The bilevel formulation is intended to reduce the coupling between structural comparison and coefficient calibration; the comparison does not establish that the gain arises from program representation, operator choice, or expression size.

Bi-EZP also obtains higher rank correlations than the reported AZ-NAS and EZNAS-A results. These comparisons establish an empirical difference under the reported protocol,

TABLE I
SCORE-ACCURACY CORRELATION COMPARISON ON NATS-BENCH.

Method	NATS-Bench-TSS			NATS-Bench-SSS		
	C-10	C-100	IN-16	C-10	C-100	IN-16
ZiCo [14]	0.604	0.593	0.589	0.709	0.536	0.719
Zen-Score [31]	0.236	0.236	0.277	0.500	0.520	0.690
SWAP [32]	0.641	0.664	0.613	0.502	0.233	0.430
GradNorm [15]	0.466	0.474	0.429	0.524	0.338	0.512
SynFlow [47]	0.581	0.568	0.561	0.610	0.600	0.390
ePADS [48]	0.641	0.652	0.620	0.384	0.137	0.351
NASWOT [12]	0.604	0.622	0.601	0.430	0.184	0.403
NI [49]	0.508	0.517	0.481	0.722	0.468	0.667
GradSign [50]	0.619	0.600	0.593	0.210	0.160	0.040
NTK [30]	-0.349	-0.364	-0.321	0.200	0.618	0.423
GraSP [11]	0.385	0.388	0.395	-0.090	0.010	0.290
SNIP [10]	0.472	0.474	0.433	0.420	0.460	0.570
AZ-NAS [17]	0.741	0.723	0.710	0.581	0.350	0.531
#Params	0.576	0.552	0.519	0.190	0.210	0.380
#FLOPs	0.541	0.517	0.487	0.530	0.540	0.650
MeCo [13]	0.730	0.711	0.669	-0.601	-0.717	-0.581
MeCoopt [13]	0.691	0.712	0.689	0.674	0.641	0.617
ParZC [34]	0.706	0.743	0.699	–	–	–
ξ-GSNR [51]	0.661	0.658	0.608	–	–	–
EZNAS-A [19]	0.650	0.650	0.610	–	–	–
ECP [18]	0.782	0.771	0.740	0.771	0.722	0.806
Bi-EZP	**0.809**	**0.791**	**0.775**	**0.792**	**0.733**	**0.826**

TABLE II
SCORE-ACCURACY CORRELATION COMPARISON ON NDS SEARCH SPACES.

Method	DARTS	ENAS	PNAS	NASNet	Amoeba
ZiCo [14]	0.345	0.197	0.195	0.087	-0.019
SWAP [32]	0.446	0.345	0.342	0.266	0.138
GradSign [50]	0.537	0.424	0.396	0.290	0.250
NASWOT [12]	0.480	0.387	0.363	0.299	0.208
ePADS [48]	0.507	0.485	0.429	0.404	0.374
SynFlow [47]	-0.001	-0.092	-0.090	-0.191	-0.001
EZNAS-A [19]	0.560	0.520	0.510	0.440	0.450
#FLOPs	0.500	0.413	0.395	0.288	0.238
#Params	0.493	0.411	0.387	0.289	0.241
AZ-NAS [17]	0.416	0.446	0.375	0.396	0.368
NI [49]	0.305	0.327	0.268	0.278	0.114
ξ-GSNR [51]	0.547	0.440	0.403	0.313	0.226
ParZC [34]	0.503	0.506	–	0.385	–
MeCo [13]	0.204	0.122	0.095	0.072	0.102
MeCoopt [13]	0.486	0.403	0.375	0.314	0.188
GradNorm [15]	0.227	0.055	0.109	0.080	-0.116
ECP [18]	0.568	0.512	0.476	0.468	0.406
Bi-EZP	**0.621**	**0.582**	**0.551**	**0.519**	**0.511**

TABLE III
CROSS-DATASET KENDALL'S τ OF FROZEN BI-EZP PROXIES ON NATS-BENCH-TSS. ROWS ARE DISCOVERY SOURCES AND COLUMNS ARE EVALUATION TARGETS. BOLD MARKS THE BEST SOURCE FOR EACH TARGET.

Source \ Target	C-10	C-100	IN-16
CIFAR-10	**0.809**	0.786	0.731
CIFAR-100	0.804	**0.791**	0.748
ImageNet-16-120	0.799	0.784	**0.775**

but they do not isolate program representation, operator choice, or expression size as its cause.

2) Experimental Result in Rank Correlation Across NDS Benchmarks: To further examine the structural generalization capability of the discovered proxies, we extend the evaluation to the Network Design Spaces (NDS) benchmark, which encompasses highly heterogeneous architectural families, including DARTS, ENAS, PNAS, NASNet, and Amoeba. Compared with NATS-Bench, the NDS benchmark presents significantly greater variability in architectural topology, thereby constituting a more stringent test of cross-space robustness.

As summarized in Table II, Bi-EZP obtains the highest rank correlation among the compared methods across the five evaluated architectural families. Relative to ECP, the correlation increases from 0.568 to 0.621 on DARTS, from 0.512 to 0.582 on ENAS, and from 0.406 to 0.511 on Amoeba. These per-space results establish an empirical improvement under task-specific discovery, while the frozen source–target evaluation below examines whether a discovered proxy can be reused without recalibration.

The per-space comparisons alone do not rule out search-space-specific fitting. Cross-space reuse is therefore assessed separately by freezing both the aggregation program and its calibrated parameters before transfer.

3) Cross-Dataset and Cross-Search-Space Proxy Performance: The preceding comparisons evaluate proxies discovered separately for each target. We further consider a stricter transfer setting, where the complete proxy discovered on a source task, including both its aggregation program and CMA-ES-optimized parameters, is frozen and directly evaluated on other targets. Rows in Tables III and IV denote the discovery sources, while columns denote the evaluation targets. Diagonal entries represent settings in which the discovery and evaluation tasks are the same, whereas off-diagonal entries measure transfer performance without rediscovery or recalibration.

Table III shows a consistent cross-dataset pattern. Each target achieves its highest correlation when the source dataset matches the target, but every off-diagonal transfer remains between 0.731 and 0.804. Averaged over the three matched settings, τ is 0.792; the mean over the six transferred settings is 0.775, a decrease of 0.017. Transfer is strongest between CIFAR-10 and CIFAR-100, where the two directions yield 0.804 and 0.786. Transfer to ImageNet-16-120 is more sensitive to the source: the CIFAR-100 proxy reaches 0.748, compared with 0.731 for the CIFAR-10 proxy. These results indicate that the frozen program retains most of its ranking fidelity across datasets within the same topology search space, while the remaining gap reflects a measurable source–target effect.

The NDS matrix in Table IV covers a larger structural shift. Off-diagonal correlations range from 0.429 to 0.627, with a mean of 0.515 compared with 0.557 on the diagonal. Transfer among DARTS, ENAS, and PNAS is particularly stable: their cross-space values fall between 0.519 and 0.627. The PNAS-discovered proxy attains the highest DARTS correlation (0.627), slightly exceeding the DARTS-discovered proxy (0.621), which shows that a matched source is not uniformly necessary for the best target ranking. NASNet and Amoeba expose a larger shift: proxies transferred into these targets range from 0.454 to 0.519 and from 0.429 to 0.511, respectively. These results provide evidence of cross-space reuse, while the source–target differences indicate that the proxy is not invariant to architectural-family shifts. The observed robustness may be facilitated by the separation of structural search and numerical calibration.

4) Effectiveness and Efficiency in Open-Domain Architecture Search: While rank correlation provides an indirect measure of proxy fidelity, the ultimate criterion of a proxy's practical value lies in its ability to guide end-to-end architecture search. To this end, we integrate Bi-EZP into a full neural architecture search pipeline within the continuous relaxation of the DARTS search space, thereby evaluating its effectiveness under realistic search dynamics.

As summarized in Table V, architectures discovered under the guidance of Bi-EZP reach test errors of 2.47% on CIFAR-10 and 16.10% on CIFAR-100. ECP reports 2.52% on CIFAR-10 under the same tabulated search cost. These results show that the rank-correlation improvements are accompanied by competitive downstream architecture quality under the reported protocol.

Bi-EZP-guided search is also compared with reinforcement-learning approaches such as ENAS and gradient-based methods including PC-DARTS and GDAS. The reported 0.06 GPU-days correspond only to the downstream DARTS architecture-search stage and do not include the one-time offline Bi-EZP proxy-discovery cost, which comprises LLM inference, base-proxy extraction, CMA-ES evaluations, and validation. The tabulated number should therefore be interpreted as downstream search cost rather than total method-development cost.

a) Scalability to Large-Scale Datasets.: To further evaluate the scalability of the discovered proxy beyond moderate-scale benchmarks, we extend the search process to the high-resolution ImageNet dataset. Following the established evaluation protocol in prior evolutionary proxy frameworks such as ECP [18], directly evolving task-specific proxy coefficients on ImageNet is computationally prohibitive due to the substantial evaluation overhead. Therefore, to ensure both fairness and computational feasibility, we transfer the proxy structure evolved on the DARTS-CIFAR-10 search space to the DARTS-ImageNet setting, leveraging the structural similarity between the two search domains.

Under this transfer-based protocol, the architecture discovered under Bi-EZP guidance achieves a Top-1 test error of 24.7% with 5.3M parameters, as reported in Table VI. The corresponding downstream architecture-search stage costs

TABLE IV
CROSS-SEARCH-SPACE KENDALL'S τ OF FROZEN BI-EZP PROXIES ON NDS. ROWS ARE DISCOVERY SOURCES AND COLUMNS ARE EVALUATION TARGETS. BOLD MARKS THE BEST SOURCE FOR EACH TARGET.

Source \ Target	DARTS	ENAS	PNAS	NASNet	Amoeba
DARTS	0.621	0.558	0.519	0.486	0.476
ENAS	0.625	**0.582**	0.548	0.488	0.429
PNAS	**0.627**	0.563	**0.551**	0.485	0.434
NASNet	0.528	0.501	0.479	**0.519**	0.479
Amoeba	0.549	0.554	0.519	0.454	**0.511**

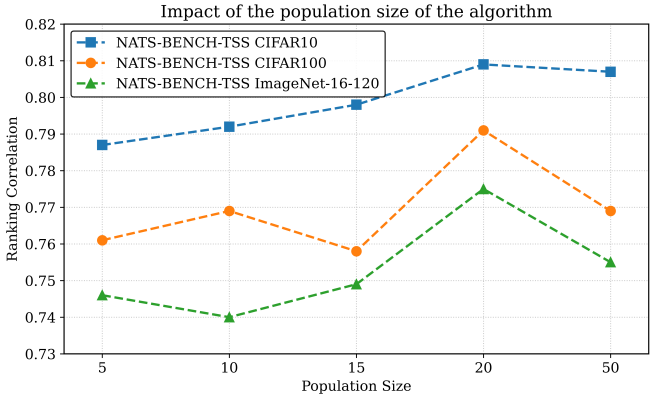

Fig. 2. Sensitivity of Bi-EZP to population size on NATS-Bench-TSS.

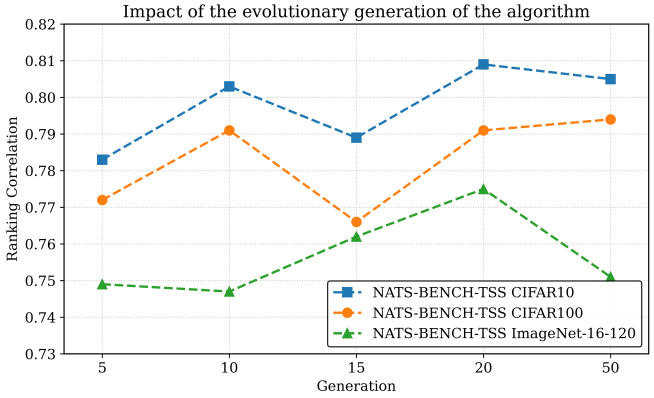

Fig. 3. Sensitivity of Bi-EZP to the number of evolutionary generations on NATS-Bench-TSS.

0.07 GPU-days; as above, this value excludes the one-time offline proxy-discovery cost. The result shows that a proxy discovered in the DARTS-CIFAR-10 setting can be used to guide the reported DARTS-ImageNet search, while broader transfer claims remain outside the evaluated setting.

D. Ablation Study and Hyperparameter Analysis

To understand the contribution of the main mechanisms in Bi-EZP, we conduct an ablation study on NATS-Bench (Table VII) and analyze sensitivity to population size and the number of evolutionary generations (Figs. 2 and 3). The updated sweeps evaluate both quantities up to 50, allowing us to distinguish a useful operating region from a simple monotonic budget effect.

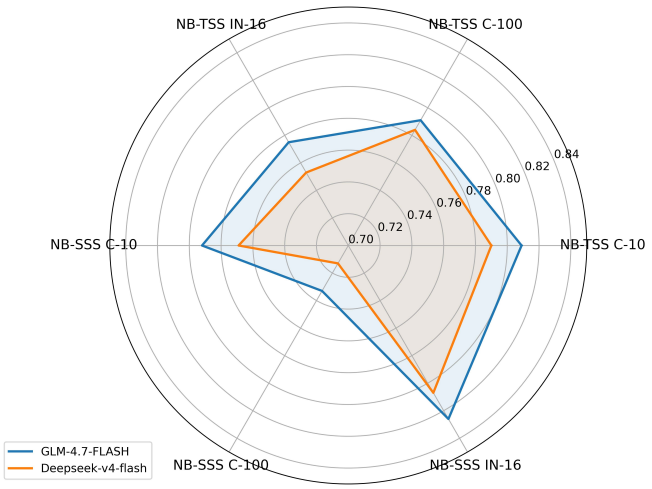

Fig. 4. Performance Comparison under Different LLMs of the proposed framework.

1) The Efficacy of Bilevel Decoupling and LLM-Driven Semantic Search: Bi-EZP uses a bilevel design to separate discrete structure proposal from continuous parameter calibration. To assess this separation, we first evaluate a variant denoted as **"w/o Bilevel Structure"**. In this configuration, the LLM directly generates both the symbolic computation graph and its numerical coefficients, collapsing the search into a single level. Table VII reports lower correlation for this variant across the evaluated datasets, including a reduction from 0.809 to 0.783 on TSS CIFAR-10. These observations are consistent with a benefit from calibrating continuous coefficients with a dedicated lower-level optimizer under the reported protocol.

Secondly, to assess the effect of the upper-level proposal mechanism, we test the **"w/o LLM (ECP Aggregation + CMA-ES)"** variant. This baseline replaces the LLM proposal operator with the ECP aggregation structure while retaining CMA-ES for lower-level calibration. It is inferior to the complete framework under the reported protocol (e.g., 0.784 versus 0.809 on TSS CIFAR-10). Following the motivation of EoH [76], we use an LLM because it can express candidate heuristics jointly as a short functional description and executable code, and can propose context-conditioned transfor-

TABLE V
COMPARISON OF BI-EZP WITH REPRESENTATIVE NAS METHODS ON CIFAR-10 AND CIFAR-100.

Architecture	Test Error (%)		Params	Search Cost	Search Strategy
	CIFAR-10	CIFAR-100	(M)	(GPU-Days)	
DenseNet [52]	3.46	17.18	25.6	-	-
ResNet [53]	4.61	22.1	1.7	-	-
VGG [54]	6.66	28.05	20.1	-	-
ENAS [3]	2.89	19.43	4.6	0.5	RL
MetaQNN [55]	6.92	17.14	-	100	RL
ADARTS [7]	2.46	17.03	2.9	0.2	GD
β-DARTS [8]	2.53	16.24	3.78	0.4	GD
PC-DARTS [56]	2.57	16.90	3.6	0.1	GD
DARTS- [57]	2.59	17.51	3.4	0.4	GD
FairDARTS [58]	2.54	17.61	2.8	0.4	GD
SNAS [59]	3.10	20.09	2.8	1.5	GD
IS-DARTS [60]	2.56	-	4.25	0.42	GD
DARTS(2st) [6]	2.76	-	3.3	1.0	GD
GDAS [61]	2.93	19.18	3.4	0.2	GD
DrNAS [62]	2.54	-	4.0	0.4	GD
DARTS+PT [63]	2.61	-	3.0	0.8	GD
DARTS(1st) [6]	3.00	17.54	3.4	0.4	GD
Cars [64]	2.62	-	3.6	0.4	EA
GENAS [65]	2.49	16.96	3.20	0.26	EA
CNN-GA (CIFAR100) [5]	-	20.53	4.1	40	EA
EAEPSO [66]	2.74	16.94	2.94	2.2	EA
AmoebaNet-A [67]	3.34	17.63	3.3	3150	EA
SLE-NAS-B [68]	3.47	18.07	0.94	0.4	EA
NPENAS-NP [69]	3.62	26.76	3.5	1.8	EA
SaDENAS [70]	2.59	16.91	3.24	0.2	EA
CNN-GA (CIFAR10) [5]	3.22	-	2.9	35	EA
EPCNAS-C [71]	3.07	18.36	1.16	1.10	EA
NSGANetV1-A2 [72]	2.65	17.42	**0.9**	27	EA
NTK [30]	2.89	20.30	4.1	0.21	ZS
GraSP [11]	2.73	22.65	3.3	0.1	ZS
MeCo [13]	2.69	16.86	4.2	0.08	ZS
SynFlow [47]	2.96	19.82	5.1	0.03	ZS
NASWOT [15]	2.77	22.90	4.8	0.06	ZS
Sweetimator [40]	2.54	-	4.6	0.05	ZS
ZiCo [14]	2.80	19.54	5.1	0.04	ZS
SNIP [10]	2.90	19.95	4.0	0.04	ZS
ECP [18]	2.52	-	4.5	**0.06**	ZS
Bi-EZP	**2.47**	**16.10**	3.3	**0.06**	ZS

TABLE VI
COMPARISON WITH REPRESENTATIVE NAS METHODS ON IMAGENET.

Architecture	Test error top-1(%)	Search cost (GPU-Days)	Params (M)	Search Strategy
ResNet [53]	30.1	-	6.6	Manual
VGG [54]	29.4	-	4.2	Manual
DenseNet [52]	29.4	-	4.2	Manual
DARTS(2st) [6]	26.7	1.0	4.7	GD
SNAS [59]	27.3	1.5	**2.8**	GD
PC-DARTS [56]	25.1	0.1	4.7	GD
GDAS [61]	26.0	0.3	3.4	GD
ProxylessNAS [73]	24.9	8.3	7.1	GD
DARTS+PT [63]	25.5	3.4	4.7	GD
AmoebaNet-B [67]	26.0	3150	5.3	EA
CARS [64]	24.8	0.4	5.1	EA
NSGANetV1-A2 [72]	25.5	27	4.1	EA
EAEPSO [66]	26.9	4.0	4.9	EA
EPCNAS-C2 [71]	27.1	1.17	3.0	EA
EG-NAS [29]	24.9	0.1	5.3	EA
LAPT-NAS [37]	24.9	2	4.6	EA
TE-NAS [30]	26.2	**0.05**	6.3	ZS
NASI-ADA [74]	24.8	0.07	5.2	ZS
QE-NAS [75]	25.5	0.02	3.2	ZS
ECP [18]	**24.7**	0.07	6.7	ZS
Bi-EZP	**24.7**	0.07	5.3	ZS

mations without requiring us to enumerate a closed grammar of all admissible compositions. Selection remains empirical: generated programs are retained only after interface validation, CMA-ES calibration, and validation-set rank-correlation evaluation. Since we do not report matched expression-size or search-efficiency measurements for a GP system, we draw no comparative conclusion about expression size or interpretability.

2) Comparison of Lower-Level Optimizers: Once the upper-level LLM formulates the explicit parameter boundaries, the lower-level solver must navigate this continuous manifold to maximize the Kendall rank correlation strictly on the inner-training set. Table VII presents a comparative analysis of different continuous optimizers, replacing CMA-ES with Genetic Algorithm (GA), Differential Evolution (DE), and Particle Swarm Optimization (PSO).

Under the reported settings, **Bi-EZP (CMA-ES)** obtains higher rank correlation than the GA-, DE-, and PSO-based variants across the evaluated search spaces. We use CMA-ES because its derivative-free covariance adaptation is compatible with the non-differentiable rank-correlation objective and can model dependencies among continuous parameters. This ablation is limited to the three alternative optimizers in Table VII

TABLE VII
ABLATION STUDY OF DIFFERENT COMPONENTS IN BI-EZP ON NATS-BENCH.

Method	NATS-Bench-TSS			NATS-Bench-SSS		
	C-10	C-100	IN-16	C-10	C-100	IN-16
w/o Bilevel Structure	0.783	0.741	0.731	0.743	0.678	0.806
w/o LLM (ECP Aggregation + CMA-ES)	0.784	0.771	0.740	0.769	0.656	0.805
Bi-EZP (GA)	0.807	0.783	0.737	0.751	0.725	0.580
Bi-EZP (DE)	0.788	0.771	0.752	0.768	0.713	0.805
Bi-EZP (PSO)	0.789	0.774	0.756	0.780	0.719	0.798
Bi-EZP	**0.809**	**0.791**	**0.775**	**0.792**	**0.733**	**0.826**

and does not establish that CMA-ES is optimal among all possible lower-level solvers.

3) Impact of Evolutionary Hyperparameters: We evaluate population sizes and generation budgets in $\{5, 10, 15, 20, 50\}$, as shown in Figs. 2 and 3. Each point reports the final Kendall correlation for one configuration; no uncertainty interval is available, so the analysis concerns the observed sensitivity rather than statistical monotonicity.

For population size, performance generally improves from $P = 5$ to $P = 20$. On CIFAR-10, τ rises from approximately 0.787 to 0.809; CIFAR-100 improves from 0.761 to 0.791; and ImageNet-16-120 improves from 0.746 to 0.775, despite smaller non-monotonic changes at intermediate sizes. Expanding the population to $P = 50$ does not yield a further gain: CIFAR-10 remains close at about 0.807, while CIFAR-100 and ImageNet-16-120 fall to about 0.769 and 0.755. Thus, additional candidates beyond 20 increase the number of LLM generations and CMA-ES calibrations without improving the observed aggregate outcome. We use $P = 20$ as the best tested balance across the three tasks.

The generation sweep is also non-monotonic. Increasing the budget from 5 to 20 raises the observed correlation from about $0.783/0.772/0.749$ to $0.809/0.791/0.775$ on CIFAR-10, CIFAR-100, and ImageNet-16-120, respectively, with a common dip at 15 generations. Extending the run to 50 generations produces mixed results: CIFAR-100 increases slightly to about 0.794, whereas CIFAR-10 declines to about 0.805 and ImageNet-16-120 to about 0.751. The extra generations therefore do not provide a consistent cross-dataset benefit. We retain $T_{max} = 20$ because it gives the strongest balanced result among the tested budgets and requires less than half the candidate evaluations of the 50-generation setting. These curves support a practical budget choice, but they do not by themselves establish convergence or identify the causes of the intermediate fluctuations.

4) Performance Sensitivity under Different LLMs: To examine sensitivity to the upper-level generative backbone, we compare GLM-4.7-Flash [46] and DeepSeek-V4-Flash [77]. Figure 4 reports competitive rank correlations for both variants across the evaluated benchmarks, with GLM-4.7-Flash producing higher correlations in several settings, including ImageNet-16-120 under TSS and SSS. This two-model comparison shows that the evaluated procedure can operate with either tested backbone, but it does not establish invariance across LLM providers, model scales, or decoding configurations.

E. Evolutionary Trajectory and Search Dynamics

To elucidate the search dynamics of the proposed framework, we analyze the step-wise convergence of the Kendall rank correlation coefficient (K_τ) over 20 generations (Figure 5).

The trajectory shows a rapid increase in the best observed validation correlation during Generations 1-5 (K_τ reaching 0.8115), followed by a plateau during Generations 5-16 and a later improvement. This trace describes the best-so-far fitness in one run; it does not reveal why individual proposals were rejected and should not be interpreted as evidence of global convergence. The best-so-far correlation reaches 0.8255 at Generation 17 and remains unchanged through the recorded budget.

V. CONCLUSION

Bi-EZP separates LLM-based program proposal from CMA-ES-based continuous parameter calibration for automated ensemble zero-shot proxy discovery. Under the evaluated protocols, the resulting proxies achieve higher rank correlation than the compared baselines on NATS-Bench and NDS and can be used as evaluation signals in downstream DARTS search.

Despite its efficacy, the current framework exhibits certain limitations. The offline proxy discovery phase introduces nontrivial token costs and latency due to iterative LLM querying. Additionally, the quality of the discovered proxy is bounded by the intrinsic properties of the pre-selected base metrics, and the structural diversity remains sensitive to the contextual alignment of prompt constraints. Future work will focus on expanding the bilevel formulation into a multi-objective paradigm to handle hardware-aware constraints like device latency.

REFERENCES

[1] X. He, K. Zhao, and X. Chu, "Automl: A survey of the state-of-the-art," *Knowledge-based systems*, vol. 212, p. 106622, 2021.
[2] B. Zoph, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.
[3] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *International conference on machine learning*. PMLR, 2018, pp. 4095–4104.
[4] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *International conference on machine learning*. PMLR, 2017, pp. 2902–2911.

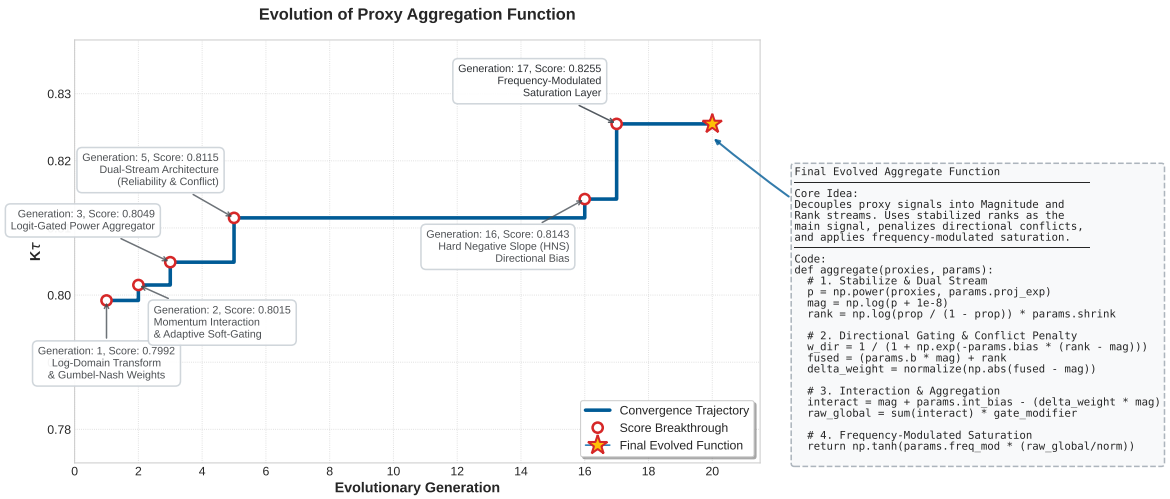

Fig. 5. The evolutionary trajectory of the proposed framework over 20 generations on CIFAR10 of the NATS-Bench TSS benchmark.

[5] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically designing cnn architectures using the genetic algorithm for image classification," *IEEE transactions on cybernetics*, vol. 50, no. 9, pp. 3840–3854, 2020.

[6] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[7] Y. Xue and J. Qin, "Partial connection based on channel attention for differentiable neural architecture search," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 5, pp. 6804–6813, 2022.

[8] P. Ye, B. Li, Y. Li, T. Chen, J. Fan, and W. Ouyang, "b-darts: Beta-decay regularization for differentiable architecture search," in *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 874–10 883.

[9] G. Li, D. Hoang, K. Bhardwaj, M. Lin, Z. Wang, and R. Marculescu, "Zero-shot neural architecture search: Challenges, solutions, and opportunities," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 12, pp. 7618–7635, 2024.

[10] N. Lee, T. Ajanthan, and P. H. Torr, "Snip: Single-shot network pruning based on connection sensitivity," *arXiv preprint arXiv:1810.02340*, 2018.

[11] C. Wang, G. Zhang, and R. Grosse, "Picking winning tickets before training by preserving gradient flow," *arXiv preprint arXiv:2002.07376*, 2020.

[12] J. Mellor, J. Turner, A. Storkey, and E. J. Crowley, "Neural architecture search without training," in *International conference on machine learning*. PMLR, 2021, pp. 7588–7598.

[13] T. Jiang, H. Wang, and R. Bie, "Meco: zero-shot nas with one data and single forward pass via minimum eigenvalue of correlation," *Advances in Neural Information Processing Systems*, vol. 36, pp. 61 020–61 047, 2023.

[14] G. Li, Y. Yang, K. Bhardwaj, and R. Marculescu, "Zico: Zero-shot nas via inverse coefficient of variation on gradients," *arXiv preprint arXiv:2301.11300*, 2023.

[15] M. S. Abdelfattah, A. Mehrotra, Ł. Dudziak, and N. D. Lane, "Zero-cost proxies for lightweight nas," *arXiv preprint arXiv:2101.08134*, 2021.

[16] X. Ning, C. Tang, W. Li, Z. Zhou, S. Liang, H. Yang, and Y. Wang, "Evaluating efficient performance estimators of neural architectures," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 265–12 277, 2021.

[17] J. Lee and B. Ham, "Az-nas: Assembling zero-cost proxies for network architecture search," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024, pp. 5893–5903.

[18] J. Huang, B. Xue, Y. Sun, and M. Zhang, "Evolving comprehensive proxies for zero-shot neural architecture search," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2025, pp. 1246–1254.

[19] Y. Akhauri, J. Munoz, N. Jain, and R. Iyer, "Eznas: evolving zero-cost proxies for neural architecture scoring," *Advances in Neural Information Processing Systems*, vol. 35, pp. 30 459–30 470, 2022.

[20] Z. Wei, P. Dong, Z. Hui, A. Li, L. Li, M. Lu, H. Pan, and D. Li, "Autoprox: Training-free vision transformer architecture search via automatic proxy discovery," in *Proceedings of the aaai conference on artificial intelligence*, vol. 38, no. 14, 2024, pp. 15 814–15 822.

[21] Q. M. Phan and N. H. Luong, "From hand-crafted metrics to evolved training-free performance predictors for neural architecture search via symbolic regression," *Neurocomputing*, p. 132440, 2025.

[22] Z. Ma, Y.-J. Gong, H. Guo, J. Chen, Y. Ma, Z. Cao, and J. Zhang, "Llamoco: Instruction tuning of large language models for optimization code generation," *IEEE Transactions on Evolutionary Computation*, 2026.

[23] C. Wang, S. Ma, Z. Ma, and Y.-J. Gong, "Evolution of benchmark: Black-box optimization benchmark design through large language model," *arXiv preprint arXiv:2601.21877*, 2026.

[24] B. Huang, X. Wu, Y. Zhou, J. Wu, L. Feng, R. Cheng, and K. C. Tan, "Evaluation of large language models as solution generators in complex optimization," *IEEE Computational Intelligence Magazine*, vol. 20, no. 4, pp. 56–70, 2025.

[25] M. Zheng, X. Su, S. You, F. Wang, C. Qian, C. Xu, and S. Albanie, "Can gpt-4 perform neural architecture search?" *arXiv preprint arXiv:2304.10970*, 2023.

[26] A. Chen, D. Dohan, and D. So, "Evoprompting: Language models for code-level neural architecture search," *Advances in neural information processing systems*, vol. 36, pp. 7787–7817, 2023.

[27] M. U. Nasir, S. Earle, J. Togelius, S. James, and C. Cleghorn, "Llmatic: neural architecture search via large language models and quality diversity optimization," in *proceedings of the Genetic and Evolutionary Computation Conference*, 2024, pp. 1110–1118.

[28] Z. Cai, Y. Tang, Y. Lai, H. Wang, Z. Chen, and H. Chen, "Seki: Self-evolution and knowledge inspiration based neural architecture search via large language models," *arXiv preprint arXiv:2502.20422*, 2025.

[29] Z. Cai, L. Chen, P. Liu, T. Ling, and Y. Lai, "Eg-nas: Neural architecture search with fast evolutionary exploration," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 38, no. 10, 2024, pp. 11 159–11 167.

[30] W. Chen, X. Gong, and Z. Wang, "Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective," *arXiv preprint arXiv:2102.11535*, 2021.

[31] M. Lin, P. Wang, Z. Sun, H. Chen, X. Sun, Q. Qian, H. Li, and R. Jin, "Zen-nas: A zero-shot nas for high-performance image recognition," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 347–356.

[32] Y. Peng, A. Song, H. M. Fayek, V. Ciesielski, and X. Chang, "Swap-nas: Sample-wise activation patterns for ultra-fast nas," *arXiv preprint arXiv:2403.04161*, 2024.

[33] S. Casarin, S. Escalera, and O. Lanz, "L-swag: Layer-sample wise activation with gradients information for zero-shot nas on vision transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025, pp. 4441–4451.

[34] P. Dong, L. Li, Z. Tang, X. Liu, Z. Wei, Q. Wang, and X. Chu, "Parzc: Parametric zero-cost proxies for efficient nas," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 15, 2025, pp. 16 327–16 335.

[35] M. Lin and J. Luo, "Per-architecture training-free metric optimization for neural architecture search," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[36] J. Huang, B. Xue, Y. Sun, M. Zhang, and G. G. Yen, "Evorep: Evolving reliable ensemble of proxies for zero-shot neural architecture search," *IEEE Transactions on Evolutionary Computation*, 2025.

[37] X. Zhou, X. Wu, L. Feng, Z. Lu, and K. C. Tan, "Design principle transfer in neural architecture search via large language models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 21, 2025, pp. 23 000–23 008.

[38] Z. Ji, G. Zhu, C. Yuan, and Y. Huang, "Rz-nas: Enhancing llm-guided neural architecture search via reflective zero-cost strategy," in *Forty-second International Conference on Machine Learning*, 2025.

[39] Y. Lai, Z. Cai, L. Chen, T. Ling, and H.-L. Liu, "Llmenas: Evolutionary neural architecture search via large language model guidance," *IEEE Transactions on Evolutionary Computation*, 2026.

[40] L. Yang, Y. Fu, S. Lu, Z. Sun, J. Mei, W. Zhao, and Y. Hu, "Sweet gradient matters: Designing consistent and efficient estimator for zero-shot architecture search," *Neural Networks*, vol. 168, pp. 237–255, 2023.

[41] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es)," *Evolutionary computation*, vol. 11, no. 1, pp. 1–18, 2003.

[42] X. Dong, L. Liu, K. Musial, and B. Gabrys, "Nats-bench: Benchmarking nas algorithms for architecture topology and size," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 7, pp. 3634–3646, 2021.

[43] I. Radosavovic, J. Johnson, S. Xie, W.-Y. Lo, and P. Dollár, "On network design spaces for visual recognition," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1882–1890.

[44] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[45] L. Xiang, L. Dudziak, M. S. Abdelfattah, T. Chau, N. D. Lane, and H. Wen, "Zero-cost operation scoring in differentiable architecture search," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 37, no. 9, 2023, pp. 10 453–10 463.

[46] T. Glm, A. Zeng, B. Xu, B. Wang, C. Zhang, D. Yin, D. Zhang, D. Rojas, G. Feng, H. Zhao *et al.*, "Chatglm: A family of large language models from glm-130b to glm-4 all tools," *arXiv preprint arXiv:2406.12793*, 2024.

[47] H. Tanaka, D. Kunin, D. L. Yamins, and S. Ganguli, "Pruning neural networks without any data by iteratively conserving synaptic flow," *Advances in neural information processing systems*, vol. 33, pp. 6377–6389, 2020.

[48] J. Huang, B. Xue, Y. Sun, M. Zhang, and G. G. Yen, "Efficient perturbation-aware distinguishing score for zero-shot neural architecture search," *Applied Soft Computing*, p. 113447, 2025.

[49] M.-T. Wu, H.-I. Lin, and C.-W. Tsai, "A training-free neural architecture search algorithm based on search economics," *IEEE Transactions on Evolutionary Computation*, vol. 28, no. 2, pp. 445–459, 2023.

[50] Z. Zhang and Z. Jia, "Gradsign: Model performance inference with theoretical insights," *arXiv preprint arXiv:2110.08616*, 2021.

[51] Z. Sun, Y. Sun, L. Yang, S. Lu, J. Mei, W. Zhao, and Y. Hu, "Unleashing the power of gradient signal-to-noise ratio for zero-shot nas," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 5763–5773.

[52] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[53] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[54] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[55] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," *arXiv preprint arXiv:1611.02167*, 2016.

[56] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, and H. Xiong, "Pc-darts: Partial channel connections for memory-efficient architecture search," *arXiv preprint arXiv:1907.05737*, 2019.

[57] X. Chu, X. Wang, B. Zhang, S. Lu, X. Wei, and J. Yan, "Darts-: Robustly stepping out of performance collapse without indicators," in *International Conference on Learning Representations*.

[58] X. Chu, T. Zhou, B. Zhang, and J. Li, "Fair darts: Eliminating unfair advantages in differentiable architecture search," in *European conference on computer vision*. Springer, 2020, pp. 465–480.

[59] S. Xie, H. Zheng, C. Liu, and L. Lin, "Snas: stochastic neural architecture search," *arXiv preprint arXiv:1812.09926*, 2018.

[60] H. He, L. Liu, H. Zhang, and N. Zheng, "Is-darts: stabilizing darts through precise measurement on candidate importance," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 11, 2024, pp. 12 367–12 375.

[61] X. Dong and Y. Yang, "Searching for a robust neural architecture in four gpu hours," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 1761–1770.

[62] X. Chen, R. Wang, M. Cheng, X. Tang, and C.-J. Hsieh, "Drnas: Dirichlet neural architecture search," *arXiv preprint arXiv:2006.10355*, 2020.

[63] R. Wang, M. Cheng, X. Chen, X. Tang, and C.-J. Hsieh, "Rethinking architecture selection in differentiable nas," *arXiv preprint arXiv:2108.04392*, 2021.

[64] Z. Yang, Y. Wang, X. Chen, B. Shi, C. Xu, C. Xu, Q. Tian, and C. Xu, "Cars: Continuous evolution for efficient neural architecture search," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1829–1838.

[65] Y. Xue, C. Chen, and A. Słowik, "Neural architecture search based on a multi-objective evolutionary algorithm with probability stack," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 4, pp. 778–786, 2023.

[66] G. Yuan, B. Wang, B. Xue, and M. Zhang, "Particle swarm optimization for efficiently evolving deep convolutional neural networks using an autoencoder-based encoding strategy," *IEEE Transactions on Evolutionary Computation*, 2023.

[67] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the aaai conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 4780–4789.

[68] J. Huang, B. Xue, Y. Sun, M. Zhang, and G. G. Yen, "Split-level evolutionary neural architecture search with elite weight inheritance," *IEEE transactions on neural networks and learning systems*, vol. 35, no. 10, pp. 13 523–13 537, 2023.

[69] C. Wei, C. Niu, Y. Tang, Y. Wang, H. Hu, and J. Liang, "Npenas: Neural predictor guided evolution for neural architecture search," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 11, pp. 8441–8455, 2022.

[70] X. Han, Y. Xue, Z. Wang, Y. Zhang, A. Muravev, and M. Gabbouj, "Sadenas: A self-adaptive differential evolution algorithm for neural architecture search," *Swarm and Evolutionary Computation*, vol. 91, p. 101736, 2024.

[71] J. Huang, B. Xue, Y. Sun, M. Zhang, and G. G. Yen, "Particle swarm optimization for compact neural architecture search for image classification," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 5, pp. 1298–1312, 2022.

[72] Z. Lu, I. Whalen, Y. Dhebar, K. Deb, E. D. Goodman, W. Banzhaf, and V. N. Boddeti, "Multiobjective evolutionary design of deep convolutional neural networks for image classification," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 2, pp. 277–291, 2020.

[73] H. Cai, L. Zhu, and S. Han, "Proxylessnas: Direct neural architecture search on target task and hardware," *arXiv preprint arXiv:1812.00332*, 2018.

[74] Y. Shu, S. Cai, Z. Dai, B. C. Ooi, and B. K. H. Low, "Nasi: Label-and data-agnostic neural architecture search at initialization," *arXiv preprint arXiv:2109.00817*, 2021.

[75] Z. Sun, C. Ge, J. Wang, M. Lin, H. Chen, H. Li, and X. Sun, "Entropy-driven mixed-precision quantization for deep network design," *Advances in Neural Information Processing Systems*, vol. 35, pp. 21 508–21 520, 2022.

[76] F. Liu, X. Tong, M. Yuan, X. Lin, F. Luo, Z. Wang, Z. Lu, and Q. Zhang, "Evolution of heuristics: Towards efficient automatic algorithm design using large language model," in *Proceedings of the 41st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 235, 2024, pp. 32 201–32 223. [Online]. Available: https://proceedings.mlr.press/v235/liu24bs.html

[77] DeepSeek-AI, "Deepseek-v4: Towards highly efficient million-token context intelligence," *https://huggingface. co/deepseek-ai/DeepSeek-V4-Pro/blob/main/DeepSeek_V4. pdf*, 2026.