REVONAS:REFLECTIVE EVOLUTIONARY EXPLORATION FOR NEURAL ARCHITECTURE SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent progress in leveraging large language models (LLMs) has enabled Neural Architecture Search (NAS) to generate new architecture not limited from manually predefined search space. Nevertheless, LLM-driven generation remains challenging: the token-level design loop is *discrete* and *non-differentiable*, preventing constructive feedback from smoothly guiding architectural improvement. These methods, in turn, commonly suffer from mode collapse into redundant structures or drift toward infeasible designs, resulting in unnecessary generation and evaluation costs. We introduce RevoNAS, a reflective evolutionary orchestrator that effectively bridges LLM-based reasoning with feedback-aligned architectural search. First, RevoNAS presents a Multi-round Multi-expert Consensus to transfer isolated design rules into meaningful architectural clues. Then, Adaptive Reflective Exploration adjusts the degree of exploration leveraging reward variance; it explores when feedback is uncertain and refines when the improvement direction becomes sufficiently clear. Finally, Pareto-guided Evolutionary Selection effectively promotes architectures that jointly optimize accuracy, efficiency, latency, confidence, and structural diversity. On NAS-Bench-201, RevoNAS achieves state-of-the-art accuracies of 95.22%, 76.38%, and 50.72% on CIFAR-10, CIFAR-100, and ImageNet16-120, respectively, while evaluating only five candidate architectures. Further empirical analyses support its effectiveness and practical applicability.

1 INTRODUCTION

Neural Architecture Search (NAS) aims to automate the design of neural network architectures under task-specific objectives and resource constraints. However, many established NAS methods (Chen et al., 2021; Xie et al., 2018; Liu et al., 2018; Zhang et al., 2021; Falkner et al., 2018) rely on manually defined search spaces, restricting architectural exploration to a fixed set of operations and connectivity patterns. Recent NAS approaches (Yang et al., 2025) driven by large language models (LLMs) alleviate this limitation by combining natural-language reasoning with code generation to translate high-level design ideas into executable architectures. Importantly, these approaches significantly reduce reliance on fixed cell-based search spaces, enabling more flexible network design.

However, LLM-based NAS operates in a fundamentally discrete and non-differentiable regime, making it challenging to translate evaluation feedback into effective update directions for subsequent architectural revisions. Existing approaches (Nasir et al., 2024; Ji et al., 2025; Yang et al., 2025) can repeatedly revisit structurally similar designs or generate invalid candidates, compromising search diversity and architectural validity. Above all, such redundant or unsuccessful trials limit performance gains with these approaches requiring up to 210.73% more GPU hours and 125.27% more tokens than RevoNAS, as shown in Fig. 1. To tackle these limitations, we aim to bridge LLM-based reasoning with a stable, feedback-aligned evolutionary framework that encourages scalable and efficient architectural exploration while preserving architectural validity.

In this paper, we introduce RevoNAS, an LLM-based NAS framework that guides reflective exploration through evolutionary multi-agent collaboration. RevoNAS consists of three main modules: Multi-round Multi-expert Consensus (MMC), Adaptive Reflective Design Exploration (ARDE), and Pareto-guided Evolutionary Selection (PES). First, MMC decomposes fragmented architectural heuristics into conceptual sub-axes, where specialized agents collaboratively refine them into coherent and interpretable structural reasoning. Specifically, MMC allows multiple agents to iteratively

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

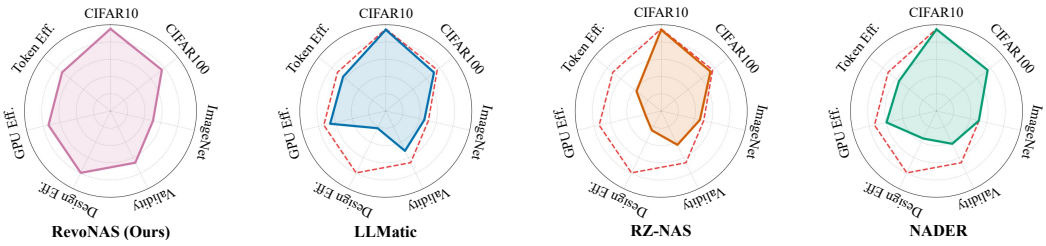


Figure 1: Comprehensive performance and efficiency comparison. All axes are displayed on a $[0, 1]$ scale, with higher values indicating better performance or efficiency. Task accuracies and validity are shown as proportions, while design, GPU, and token efficiencies are derived from normalized costs. RevoNAS demonstrates the most robust trade-off, achieving state-of-the-art accuracy with significantly lower resource consumption. Red dashed lines denote RevoNAS.

discuss and distill local architectural clues into practical drop-in knowledge, converging toward a stable consensus and a quality plateau. The design is further guided by ARDE, which effectively adjusts exploration–exploitation intensity based on reward variance (Tokic, 2010; Rodrigues Gomes & Kowalczyk, 2009; Dabney et al., 2020). In particular, it encourages exploration when reward variance is low and reflectively exploits promising design proposals as reward variance increases. Reflective summaries then update contextual memory with feedback from both successful and failed evaluations, guiding subsequent prompts to build on effective design choices and avoid repeating structural errors. As a final step, PES prioritizes resulting architectures that jointly optimize accuracy, efficiency, latency, confidence, and structural diversity, effectively discouraging selection from concentrating on a narrow family of architectures. Subsequent generations are in turn grounded in these selected architectures, which establish well-motivated design principles for reflective evolutionary development. Taken together, RevoNAS establishes a feedback-driven evolutionary loop that integrates effective architectural guidance with adaptive exploration and multi-objective parent selection, enabling well-aligned architectural design beyond manually defined search spaces.

In summary, RevoNAS effectively bridges reflective architectural reasoning with Pareto-guided evolution, distilling plausible design knowledge from evaluation feedback to drive iterative architectural refinements and reduce redundant search. On NAS-Bench-201, RevoNAS achieves 95.22%, 76.38%, and 50.72% test accuracy on CIFAR-10, CIFAR-100, and ImageNet16-120, respectively, with only 5 architecture evaluations per dataset (*i.e.*, efficiently reducing the required computational budget). Additional experiments on various downstream computer vision tasks assess its applicability across search settings, while ablation and transfer studies examine component effectiveness and transferability. Here, the key contributions of this work are as follows:

- We introduce a Multi-round Multi-expert Consensus mechanism that converts fragmented design heuristics into informative, effective architectural inspiration tokens.
- We propose an Adaptive Reflective Exploration that modulates LLM-driven search behavior via reward variance, balancing exploration-exploitation in a principled manner.
- We develop a Pareto-guided Evolutionary Selection mechanism that preserves design diversity while promoting practically strong architectures across accuracy, efficiency, and broader structural robustness metrics.
- We achieve state-of-the-art accuracy on NAS-Bench-201 with a reduced computational budget of only five architecture evaluations per dataset, supported by further empirical analyses of component effectiveness, and practicality across search settings.

2 Related Works

2.1 Neural Architecture Search

Conventional Neural Architecture Search (NAS) identifies high-performing architectures within manually defined search spaces through reinforcement learning, evolutionary algorithms, and Bayesian optimization (Tian et al., 2020; Real et al., 2019; Falkner et al., 2018). Weight-sharing and differentiable approaches improve search efficiency by reusing parameters across candidate architectures (Liu

et al., 2019; Guo et al., 2020; Xie et al., 2018; Xu et al., 2019; Zhang et al., 2021), while retaining predefined operations and connectivity patterns. Recent LLM-driven NAS methods employ natural-language reasoning and code generation to propose and refine architectures. GENIUS (Zheng et al., 2023) uses GPT-4 as a black-box optimizer to iteratively improve candidates within a specified search space. EvoPrompting (Chen et al., 2023) incorporates code-level mutation and crossover into evolutionary search, while LLMatic (Nasir et al., 2024) combines LLM-based architecture generation with quality-diversity optimization. RZ-NAS (Ji et al., 2025) integrates reflection with zero-cost evaluation to guide architectural revisions. NADER (Yang et al., 2025) further explores architectural design beyond predefined search spaces through specialized LLM agents, graph-based representations, and reflection on previous design experiences. These developments highlight the potential of LLMs for flexible architectural exploration, while motivating more structured coordination between design reasoning, and evaluation feedback. RevoNAS aims to addresses this direction through multi-round expert consensus, reward-variance-based exploration control, and Pareto-guided parent selection.

2.2 Large Language Model and Multi-Agent

Large Language Models (LLMs) have evolved into general purpose agents capable of planning, tool use, and self reflection (Shinn et al., 2023; Zhao et al., 2024a). Systems such as Visual ChatGPT (Wu et al., 2023) and HuggingGPT (Shen et al., 2023) orchestrate multiple models under LLM control to execute complex multi step workflows (Wu et al., 2023; Shen et al., 2023). A parallel line of work studies multi agent LLM collaboration, including role playing with specialized roles and decomposed tasks (Yu et al., 2024; Zhang et al., 2025b; He et al., 2025; Zhang et al., 2025a; Chen et al., 2024), debate and voting schemes that independently propose and then reconcile solutions (Kaesberg et al., 2025; Choi et al., 2025; Feng et al., 2024; Chen et al., 2025; Park et al., 2024; Liang et al., 2023), and group discussion or negotiation style protocols (Zhao et al., 2024b; Liu et al., 2025; Wang et al., 2025; Doğru et al., 2025; Nan et al., 2023; Kim et al., 2024). Recent studies highlight that adapting collaboration strategies to task complexity is increasingly crucial in high stakes settings. RevoNAS builds on these ideas, but applies multi-agent collaboration to neural architecture search, where agents operate over structured architecture representations, coordinating over longer design trajectories.

3 Methods

3.1 Overview

We formulate RevoNAS, a reflective evolutionary exploration framework for NAS (see Fig. 2), as an architectural design orchestrator that bridges LLM-driven constructive reasoning with structured exploration over the architecture space $\mathcal{A} = \{a^{(n)}\}$. To this aim, MMC (Sec. 3.2) extracts a memory of design cues ("*inspirations*") $\mathcal{I}_t = \{i_t^{(n)}\}$ from a collection of authentic papers P through a collaborative discussion process. These inspirations provide reusable structural tokens that guide architectural transformations. Building on this, the exploration module explores informative inspiration $i_t^{(n)}$ by applying transformations to either the initial base structure (at $t = 1$) or a successful parent $a_{t-1}^{(n)}$ from the previous iteration, resulting in $a_t = \mathcal{T}(\cdot, i_t^{(n)})$. Meanwhile, an exploration–exploitation controller regulates redundancy and novelty (Sec. 3.3). Each candidate is trained under a uniform protocol (Dong & Yang, 2020), and we evaluate it using multiple interpretable objectives. Finally, Pareto-based selection (Sec. 3.4) yields the optimal parent design set $\mathcal{A}_t$ for the next generation $t + 1$, while the utilities of inspirations $\mathcal{I}_t$ are updated to encourage effective architectural search, thereby completing one evolutionary cycle.

3.2 Multi-expert Multi-round Consensus

LLMs are considered to be inherently versatile: *they can interpret ideas, reorganize reasoning steps, and adapt to new conceptual spaces*. Especially, this capability suggests that LLMs, in principle, serve as powerful engines for model-based design synthesis (*i.e.*, Neural Architecture Search (Yang et al., 2025; Nasir et al., 2024; Liu et al., 2018; Chen et al., 2021; Rahman & Chakraborty, 2024)). Nevertheless, we observe that this approach typically extracts a flat set of generations from a single perspective, failing to capture how design concepts are functionally or logically connected. This is mainly because prior methods (Yang et al., 2025; Nasir et al., 2024; Zheng et al., 2023) often lack

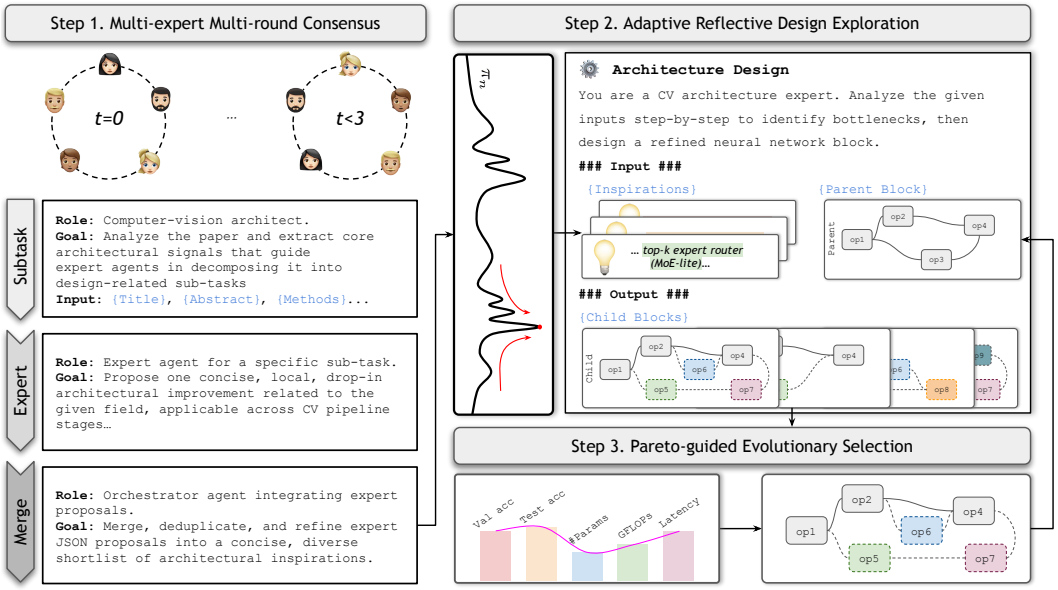


Figure 2: RevoNAS Orchestrator. Starting from a base model, a multi-expert ideation module distills literature into architectural inspirations, which are then fed to an LLM-based reflective design exploration module that proposes new candidate architectures. The candidates are trained and evaluated, and a Pareto-guided evolution preserves diverse high-quality models while updating survivors for the next generation.

effective interactive mechanisms for refinement and verification, allowing structural inconsistencies and poorly motivated design choices to persist in the resulting architectures.

To tackle this limitation, we introduce a Multi-expert Multi-round Consensus (MMC), which converts isolated design rules into global structural information. We decompose literature into distinct architecture-influential axes automatically derived from our extraction prompt. Each expert specializes in one conceptual dimension (*e.g.*, hierarchical scaling, spatial–channel decoupling) and then formulates concise, drop-in architectural signals such as `DWConv→SE→MLP` or `token gating with deformable shift`. It is noteworthy that this decomposition can transfer individual architectural reasoning into interpretable global reasoning processes, enabling more precise and modular ideations. Subsequently, we design a multi-round expert-consensus ideation that enables iterative refinement and convergence of individually proposed architectural ideas. Rather than relying on a single-shot extraction, our framework organizes structured discussions among experts to jointly encourage inspirations over multiple rounds. Precisely, given a paper P and its sub-axes $\mathcal{X} = \{x_1, \ldots, x_K\}$, each expert E_k (associated with x_k) proposes a local design clue $p_k^{(t)} = E_k(P, \mathcal{I}^{(t-1)})$ at round t, where $\mathcal{I}^{(t-1)}$ represents the shared inspiration set from the previous round. The consensus module Ω then integrates these proposals into an updated set $\mathcal{I}^{(t)} = \Omega(\{p_k^{(t)}\}_{k=1}^K)$, effectively promoting complementarity and reducing redundancy. Here, the discussion round continues until both *consensus stability* and *quality plateau* are reached:

$$\text{Jaccard}(\mathcal{I}^{(t)}, \mathcal{I}^{(t-1)}) \geq \tau_J, \quad \text{and} \quad \Delta_\mu^{(t)} \triangleq \frac{|\mu^{(t)} - \mu^{(t-1)}|}{\mu^{(t-1)}} \leq \tau_Q, \tag{1}$$

where $\mu^{(t)}$ denotes the mean quality score of inspirations at round t, and τ_J and τ_Q specify the tolerance levels. This adaptive termination ensures that ideation halts only when the architectural reasoning has converged, producing a stable yet diverse set of design primitives. In practice, this expert-consensus formulation leads to more constructive and informative inspirations $\mathcal{I}$ compared to a single-shot extraction, effectively bridging localized architectural signals.

3.3 Adaptive Reflective Design Exploration

Typical sampling heuristics in LLMs (*e.g.*, top-k, nucleus, or temperature-based decoding) implicitly trade diversity for consistency, but they remain agnostic to empirical feedback from previous designs.

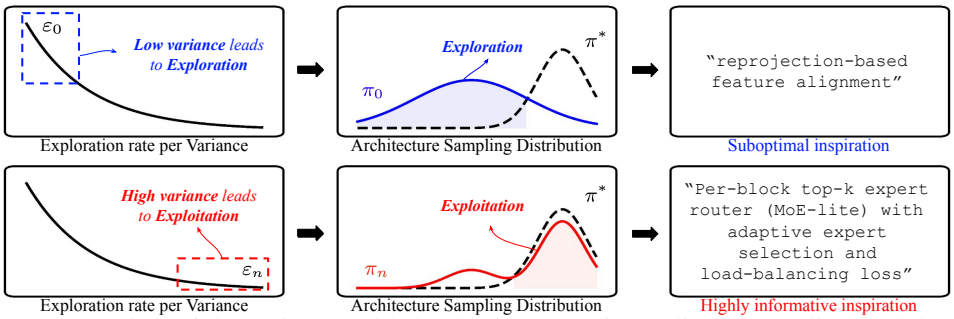


Figure 3: Overview of the variance-guided Adaptive Sampling Policy. The policy modulates the exploration rate (ε_n) inversely to the observed variance. Low variance encourages broad exploration for meaningful inspirations, whereas high variance concentrates on optimal inspirations.

In addition, designing neural architectures through LLMs entails an inherently *non-differentiable* and *discrete* search space. Specifically, unlike gradient-based NAS (Liu et al., 2018; Zhang et al., 2021; Liu et al., 2019), LLM-driven design generation lacks continuous feedback, which often leads to *mode collapse* toward over-represented and memorized architectural patterns. To overcome this limitation, we propose an Adaptive Reflective Design Exploration strategy that adaptively calibrates exploration and exploitation in the non-differentiable, knowledge-driven design space of LLMs.

We aim to explicitly regulates exploration-exploitation through an adaptive ε-greedy schedule (Tokic, 2010; Rodrigues Gomes & Kowalczyk, 2009; Dabney et al., 2020) that dynamically responds to the uncertainty of recent rewards. We reinterpret this idea in a language-conditioned design space, coupling reflective reasoning with an adaptive ε-greedy controller. This mechanism allows the agent to self-regulate its curiosity: it increases exploration when joint dynamics are unstable and exploits accumulated knowledge when the search stabilizes, as shown in Fig. 3. Let $\mathcal{M}_n = \{(i_{n-1}, r_{n-1})\}$ denote the replay memory containing prior architecture inspiration i_{n-1} and their corresponding scalar rewards r_{n-1}. Here, the reward r is computed as the performance gain, defined as the difference in accuracy between the child and its parent. For a given inspiration i, we collect the index set $\mathcal{J}(i)$ and define its reflection-based Q-value as the empirical mean reward:

$$\mathcal{J}(i) \;=\; \{\, n \mid i^{(n)} = i \,\}, \qquad \text{and} \qquad Q(i) \;=\; \begin{cases} \dfrac{1}{|\mathcal{J}(i)|} \displaystyle\sum_{n \in \mathcal{J}(i)} r^{(n)}, & \text{if } |\mathcal{J}(i)| > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

These Q-values are then used to bias inspiration sampling in ARDE, favoring inspirations that consistently yield positive improvements. To dynamically regulate this process, the exploration coefficient ε_n at each step n is defined as follows:

$$\varepsilon_n = \varepsilon_{\min} + (\varepsilon_{\max} - \varepsilon_{\min}) \cdot e^{-\lambda \cdot \mathrm{Var}(r_{n-m:n})}, \tag{3}$$

where $\mathrm{Var}(r_{n-m:n})$ measures the variance of recent rewards within the latest window $[n-m, n]$. Precisely, a lower reward variance encourages *exploration*, while a higher variance signals convergence and induces *exploitation*. This dynamic scheduling allows the search policy to adaptively modulate its exploratory intensity without hand-crafted tuning. Given the current reflective state s_n, which summarizes textual memory, prior performance, and contextual embeddings, the agent effectively samples architectural proposals according to the following:

$$\pi_n(i \mid s_n) = \begin{cases} \arg\max_i Q(i) & \text{with probability } 1 - \varepsilon_n, \\ \mathcal{A}_{\text{LLM}}(\text{prompt}(s_n, \text{Reflect}(s_{n-1}, r_{n-1}))) & \text{with probability } \varepsilon_n, \end{cases} \tag{4}$$

where $Q(i)$ denotes the empirical mean reward for a candidate inspiration i under the current reflective state s_n, and $\text{Reflect}(\cdot)$ reformulates the design prompt based on prior reasoning and performance feedback. This mechanism allows the system to alternate between exploiting historically strong design cues and exploring new reasoning paths guided by reflective memory. After evaluation, each structural inspiration i_n receives a scalar reward r_n, and the tuple (i_n, r_n) is appended to $\mathcal{M}_{n+1}$. Reflect then summarizes accumulated outcomes and failures to reshape subsequent reasoning trajectories. In this way, the agent performs implicit policy improvement without gradient updates; this reflective feedback protocol enhances representational diversity and conceptual depth, while our adaptive module enforces interpretable and variance-regulated exploration.

Algorithm 1: Pareto-guided Evolutionary Selection

Input: Candidates $\mathcal{A}_t = \{a_t^{(n)}\}$, selection weights $\{\lambda_p, \lambda_\ell, \gamma_d, \rho_c, \beta\}$, survival ratio $\kappa \in (0, 1]$
Output: Survivors $\mathcal{S}$ for the next generation

Procedure:
`# Decompose candidates using fast non-dominated sorting`
$\{\mathcal{F}_0, \mathcal{F}_1, \dots\} \leftarrow \mathrm{FastNonDominatedSort}(\mathcal{A}_t)$
$\mathcal{C} \leftarrow \begin{cases} \mathcal{F}_0, & \text{if } |\mathcal{F}_0| > 0, \\ \mathcal{A}_t, & \text{otherwise} \end{cases}$
`# Compute the selection score for each candidate`
foreach $a \in \mathcal{C}$ **do**
  Estimate $\sigma(a)$ from intermediate training dynamics
  $\Phi_{\mathrm{PES}}(a) = \mathrm{Acc}(a) - \lambda_p \,\mathrm{Params}(a) - \lambda_\ell \,\mathrm{Latency}(a) + \gamma_d \,\mathrm{StructDiv}(a) + \rho_c \,\mathrm{Conf}(a) + \beta\sigma(a)$
`# Select the highest-scoring candidates as survivors`
$K \leftarrow \max(1, \lfloor \kappa|\mathcal{C}| \rfloor)$
$\mathcal{S} \leftarrow \mathrm{TopK}(\mathcal{C}, K;\, \mathrm{key} = \Phi_{\mathrm{PES}})$
return $\mathcal{S}$

3.4 Pareto-guided Evolutionary Selection

In evolutionary NAS, repeated selection of structurally homogeneous parents and trivial variation in their offspring can progressively restrict the search to a narrow family of architectures. Under this *genetic constraint*, the search trajectory concentrates its budget on incremental refinements of inherited structures and can lead to early convergence before adequately exploring functionally motivated architectural alternatives. Consequently, it may overlook architectures that achieve improved trade-offs among accuracy, efficiency, and robustness while satisfying diverse deployment requirements, from lightweight on-device inference to reliable performance under distributional shift. To tackle this drawback, we carefully design a Pareto-guided Evolutionary Selection strategy that leverages evolutionary priors to increase the likelihood of discovering high-quality yet diverse candidates while promoting models that achieve balanced trade-offs.

In Algorithm 1, each candidate model is evaluated according to its accuracy, parameter count, latency, structural diversity, and confidence. Accuracy, structural diversity, and confidence are treated as objectives to be maximized, whereas parameter count and latency are minimized to account for both predictive performance and practical deployability. We first apply fast non-dominated sorting to decompose the candidate population into a hierarchy of Pareto fronts. The first front contains candidates that are not dominated by any other model and thus represent promising trade-offs among the considered objectives. Therefore, we adopt this first front as the candidate set $\mathcal{C}$ for subsequent selection; if it is empty, the complete candidate population is retained as a fallback.

For each candidate in $\mathcal{C}$, PES computes a selection score, denoted by Φ_{PES}, that balances predictive accuracy against parameter and latency costs while rewarding structural diversity, confidence, and exploratory potential. The exploratory component is determined by performance uncertainty estimated from intermediate training dynamics, allowing promising but insufficiently explored candidates to remain competitive. The corresponding weights control the relative importance of efficiency, structural diversity, confidence, and exploration during selection. Candidates are then ranked according to Φ_{PES}, and the highest-scoring fraction determined by the survival ratio is transferred to the next generation, with at least one candidate retained. By combining Pareto-front filtering with score-based selection, PES stabilizes the evolutionary process and prevents degeneration toward overly specialized architecture families. It ultimately maintains a consistent supply of structurally diverse yet promising candidates while guiding the search toward efficient and generalizable architectural motifs.

4 Experiments

RevoNAS carefully bridges reflective exploration, which provides design guidance from prior evaluation feedback, with Pareto-based selection to evolve architectures that balance predictive potential and computational efficiency. We therefore aim to demonstrate that RevoNAS effectively present well-motivated architectures with fewer candidate evaluations, lower GPU costs, and reduced LLM token usage.

4.1 Experimental setup.

We follow NAS-Bench-201 (Dong & Yang, 2020) and NAS-Bench-360 training and evaluation protocols. RevoNAS adopts both ResNet-18 (He et al., 2016) and minimally structured architectures as initial baselines,

Table 1: Comparison with existing state-of-the-art NAS and NAD methods on CIFAR10, CIFAR100 and ImageNet16-120. Averaged over 4 independent search runs (*i.e.*, mean±std). † and ‡ denote the Qwen2.5 and GPT-4o backbones, respectively.

Method	# Archs	CIFAR10 (↑)		CIFAR100 (↑)		ImageNet16-120 (↑)	
		Val.	Test	Val.	Test	Val.	Test
Oracle	-	91.61	94.37	73.49	73.51	46.77	47.31
ENAS (Pham et al., 2018)	-	37.51±3.19	53.89±0.58	13.37±2.35	13.96±2.33	15.06±1.95	14.84±2.10
DARTS (Liu et al., 2018)	-	39.77±0.00	54.30±0.00	38.57±0.00	15.61±0.00	18.87±0.00	16.32±0.00
SETN (Dong & Yang, 2019a)	-	84.04±0.28	87.64±0.00	58.86±0.06	59.05±0.24	33.06±0.02	32.52±0.21
DSNAS (Hu et al., 2020)	-	89.66±0.29	93.08±0.13	30.87±16.40	31.01±16.4	40.61±0.10	41.07±0.09
PC-DARTS (Xu et al., 2019)	-	89.96±0.15	93.41±0.30	67.12±0.39	67.48±0.89	40.83±0.08	41.31±0.22
SNAS (Xie et al., 2018)	-	90.10±1.04	92.77±0.83	69.69±2.39	69.34±1.98	42.84±1.79	43.16±2.64
iDARTS (Zhang et al., 2021)	-	89.86±0.60	93.58±0.32	70.57±0.24	70.83±0.48	40.38±0.59	40.89±0.68
GDAS (Dong & Yang, 2019b)	-	89.89±0.08	93.61±0.09	71.34±0.04	70.70±0.30	41.59±1.33	41.71±0.98
DRNAS (Chen et al., 2021)	-	91.55±0.00	94.36±0.00	73.49±0.00	73.51±0.00	46.37±0.00	46.34±0.00
β-DARTS (Ye et al., 2022)	-	91.55±0.00	94.36±0.00	73.49±0.00	73.51±0.00	46.37±0.00	46.34±0.00
Λ-DARTS (Movahedi et al., 2022)	-	91.55±0.00	94.36±0.00	73.49±0.00	73.51±0.00	46.37±0.00	46.34±0.00
REA (Real et al., 2019)	500	91.19±0.31	93.92±0.30	71.81±1.12	71.84±0.99	45.15±0.89	45.54±1.03
RS (Bergstra & Bengio, 2012)	500	90.93±0.36	93.70±0.36	70.93±1.09	71.04±1.07	44.45±1.10	44.57±1.25
REINFORCE (Williams, 1992)	500	91.09±0.37	93.85±0.37	71.61±1.12	71.71±1.09	45.05±1.02	45.24±1.18
BOHB (Falkner et al., 2018)	500	90.82±0.53	93.61±0.52	70.74±1.29	70.85±1.28	44.26±1.36	44.42±1.49
GENIUS (Zheng et al., 2023)	10	91.07±0.20	93.79±0.09	70.96±0.33	70.91±0.72	45.29±0.81	44.96±1.02
LLMatic (Nasir et al., 2024)	2000	-	94.26±0.13	-	71.62±1.73	-	45.87±0.96
RZ-NAS (Ji et al., 2025))	1500	91.45±0.10	94.24±0.12	73.35±0.14	73.30±0.21	46.53±0.24	46.24±0.23
LeMo-NADe (Rahman & Chakraborty, 2024)	30	90.90	89.41	63.38	67.90	27.05	27.70
NADER (Yang et al., 2025)	5	91.17±0.24	94.52±0.22	73.29±1.86	73.12±1.09	47.98±0.73	47.99±0.38
NADER (Yang et al., 2025)	10	91.18±0.23	94.52±0.22	74.71±0.45	74.65±0.33	48.56±0.83	48.61±0.76
NADER (Yang et al., 2025)	500	91.55	94.62	75.72	76.00	50.20	50.52
RevoNAS†	5	91.17±0.25	94.77±0.18	76.21±0.42	76.25±0.37	**51.00±0.21**	50.55±0.17
RevoNAS‡	5	**92.55±0.27**	**95.22±0.20**	**76.70±0.39**	**76.38±0.32**	50.58±0.19	**50.72±0.20**

and iteratively refines it through the adaptive reflective generation strategy. For CIFAR10, CIFAR100, and ImageNet16-120, we follow the training and evaluation protocols of NAS-Bench-201 (Dong & Yang, 2020). We additionally evaluate RevoNAS on four tasks from NAS-Bench-360 and an InternImage-style 5M-parameter proxy on ImageNet-1K. We utilize various large language models, including GPT-4o, Qwen-2.5, and DeepSeek, to facilitate the architectural generation process. Additional details are provided in Appendix.

4.2 Overall Performance

Comparison with existing SOTA approaches. In this section, we quantitatively compares RevoNAS to representative NAS approaches on CIFAR10/100, and ImageNet16-120 leveraging NAS-Bench-201 (Dong & Yang, 2020) in Tab. 1. While traditional methods (*i.e.*, BOHB, REINFORCE, RS, REA) require large-scale search over hundreds of candidate architectures, and recent LLM-guided methods (*i.e.*, GENIUS, LeMo-NADe, NADER) still rely on extensive sampling or specialized retraining pipelines, RevoNAS (GPT-4o) achieves superior results with significantly fewer designed architectures. With only 5 trained architectures (*i.e.*, # Archs), RevoNAS further improves to 92.55/95.22 on CIFAR10, 76.70/76.38 on CIFAR100, and 50.58/50.72 on ImageNet16-120, establishing a new state-of-the-art for NAS under comparable or lower search cost. These results highlight that RevoNAS does not rely on large search budgets; instead, its reflective multi-expert consensus and Pareto-guided evolutionary refinement enable rapid discovery of high-quality architectures with minimal exploration overhead compared to prior methods.

Ablation Study. We quantitatively analyze the contribution of the two key components in RevoNAS: Adaptive Reflective Design Exploration (ARDE) and Pareto-guided Evolutionary Selection (PES). As shown in Tab. 2, removing both modules leads to substantially weaker performance across all datasets, indicating that naive single-shot LLM proposals are insufficient for stable architecture discovery. Introducing PES alone improves results by filtering inferior candidates through multi-objective selection, while enabling ARDE alone yields consistent gains by refining architectural motifs along concept-aligned expert axes. Combining both ARDE and PES achieves the highest performance, improving CIFAR10 test accuracy from 92.31% to 95.22%, CIFAR100 from 71.83% to 76.38%, and ImageNet16-120 from 46.90% to 50.72%. These results demonstrate that ARDE and PES are complementary and jointly essential for stable and effective LLM-guided neural architecture design.

Table 2: Ablation study of the core components in RevoNAS. The column headers indicate whether ARDE and PES are enabled, respectively.

Dataset	Split	ARDE | PES			
		✗ | ✗	✓ | ✗	✗ | ✓	✓ | ✓
CIFAR10	Val.	88.43±.37	92.09±1.28	90.61±.45	**92.55±.27**
	Test	92.31±.51	94.66±1.05	93.97±.60	**95.22±.20**
CIFAR100	Val.	71.95±.34	75.70±.44	73.16±.14	**76.70±.39**
	Test	71.83±.26	75.74±.37	73.49±.18	**76.38±.32**
ImageNet16	Val.	46.67±.50	49.60±.37	47.23±.36	**50.58±.19**
	Test	46.90±.29	49.78±.87	47.62±.34	**50.72±.20**

Table 3: Comprehensive ablation study of Pareto optimization hyperparameters across CIFAR10, CIFAR100, and ImageNet16-120 datasets. The hyperparameters λ_p, ρ_c, and λ_l denote the penalty weights for the number of parameters, model confidence, and inference latency, respectively.

Pareto Weights			CIFAR10				CIFAR100				ImageNet16-120			
λ_p	ρ_c	λ_l	# Params	GFLOPs	Latency	Acc.	# Params	GFLOPs	Latency	Acc.	# Params	GFLOPs	Latency	Acc.
0.10	0.10	0.10	1.38M	0.200	9.3ms	94.45	1.39M	0.193	23.6ms	**75.80**	1.26M	0.049	9.1ms	**50.22**
0.25	0.10	0.10	**1.28M**	**0.185**	10.7ms	94.41	**1.33M**	**0.188**	22.7ms	74.32	**1.16M**	0.049	9.0ms	48.88
0.10	0.30	0.10	1.38M	0.200	**6.0ms**	94.35	1.41M	0.193	17.6ms	73.29	1.26M	0.049	14.7ms	48.57
0.10	0.10	0.25	1.38M	0.200	9.0ms	**94.49**	1.34M	**0.188**	**16.8ms**	73.21	1.26M	0.049	**7.2ms**	49.22

4.3 EFFECTIVENESS

Interpretability via Parameter Variation. We quantitatively investigate the impact of Pareto preference weights on architecture selection to understand how different optimization priorities influence both architectural structure and deployment characteristics. Specifically, we vary the weighting coefficients for parameter cost (λ_p), confidence regularization (ρ_c), and latency constraint (λ_l), while keeping all other training conditions identical, as shown in Tab. 3. Increasing λ_p encourages more compact models with reduced computational cost, achieving minimum parameters and GFLOPs while maintaining comparable accuracy. In contrast, emphasizing latency ($\lambda_l = 0.25$) effectively reduces inference time (up to -6ms) but slightly compromises accuracy, reflecting the inherent trade-off between computational efficiency and predictive precision. It is noteworthy that while a higher confidence weight ρ_c yields only marginal accuracy gains, its efficacy in bridging the domain gap is highly encouraging, as illustrated in Fig. 5. These findings not only boost the flexible model configurations but also demonstrate substantial potential for robust cross-domain generalization. Ultimately, the ability to tune these preference weights not only enhances the interpretability of architectural design decisions (*i.e.*, clarifying how each factor contributes to performance) but also facilitates adaptive deployment across diverse computing environments, from lightweight edge devices to high-throughput GPU servers.

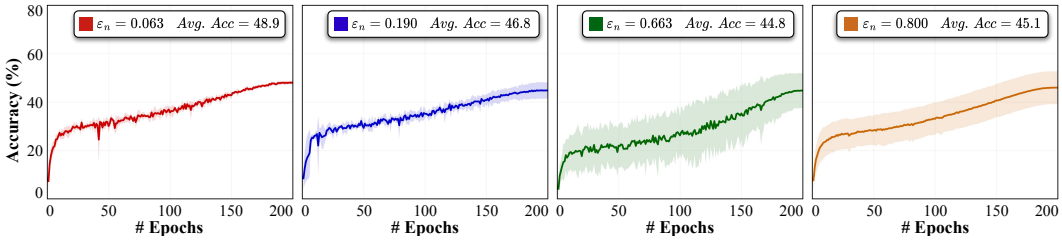


Figure 4: Adaptive exploration schedule in RevoNAS. The four panels illustrate accuracy trajectories over 200 training epochs with the corresponding ε_n. High joint dynamics of ε_n (*i.e.*, low variance) encourages exploration, while high variance drives exploitation. Note that exploration yields diverse but potentially suboptimal inspirations, whereas exploitation focuses on more effective ones.

Exploration–Variance Relationship. We showcase how exploration rate ε_n affects the performance of architectural design, as shown in Fig. 4. In particular, we analyze four joint dynamic settings in ImageNet16-120. Our results indicate that a lower reward variance (*e.g.*, triggering a high exploration rate of $\varepsilon_n = 0.800$) induces significant accuracy oscillations, reflecting a broader search space; however, the joint dynamics ε_n diminish to 0.063, yielding smoother and more stable evolutionary convergence. These findings demonstrate that maintaining an appropriate balance between exploration and reward variance leads to feedback-driven design, while facilitating a more efficient and effective architectural design space.

Evaluation on NAS-Bench-360. We evaluate RevoNAS on four tasks from NAS-Bench-360, covering spherical image classification, fluid-state prediction, sEMG classification, and genomic sequence prediction. As shown in Tab. 4, RevoNAS achieves lower errors than all evaluated non-expert baselines across the four tasks. Compared with the strongest non-expert baseline for each task, it reduces the error from 63.45% to 55.77% on Spherical and from 6.78% to 6.22% on NinaPro. It also lowers the relative ℓ_2 error from 0.016 to 0.009 on Darcy Flow and 1-AUROC from 0.37 to 0.32 on DeepSEA, approaching the corresponding expert results. These results support the applicability of RevoNAS across tasks with diverse input structures and prediction objectives.

4.4 EFFICACY

Success Rate of Pareto-guided Model Generation. While expanding search spaces in NAS boosts architectural diversity, the resulting computational overhead makes effective space reduction essential. In LLM-guided NAD, generating constructive offspring from Pareto-guided parents is crucial to avoid blind sampling. Unlike prior accuracy-centric methods like NADER, RevoNAS employs a Pareto-based evaluation—jointly assessing accuracy, efficiency, and robustness—to enable reliable evolutionary selection. As shown in Tab. 5, our strategy

Table 4: Evaluation on NAS-Bench-360. Spherical, Darcy Flow, NinaPro, and DeepSEA denote spherical image classification, fluid-state prediction, sEMG classification, and DNA chromatin/binding-state prediction, respectively.

Methods	Spherical	Darcy Flow	NinaPro	DeepSEA
	0-1 error(%)↓	relative ℓ_2↓	0-1 error(%)↓	1-AUROC↓
Expert	67.41	0.008	8.73	0.30
WRN	85.77	0.073	6.78	0.40
WRN-ASHA	75.46	0.066	7.34	0.41
DenseNAS	72.99	0.100	10.17	0.40
Perceiver IO	82.57	0.240	22.22	0.38
Baby Dash	63.45	0.016	8.28	0.37
RevoNAS	**55.77**	**0.009**	**6.22**	**0.32**

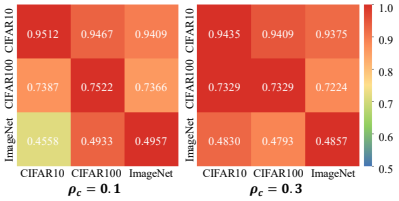


Figure 5: Cross-dataset direct transfer across confidence weights ρ_c (0.1 vs. 0.3). Each heatmap reports the test accuracy and deviation when transferring architectures from *source* (rows) to *target* (columns).

Table 5: Success rate and trial efficiency in parent-child model generation. The table details the frequency of performance gains (SR, Success Rate), average attempts to first success (Trials/1st-Succ.), and mean trials per successful model (Trials/Succ.).

Method	SR↑	Trials/1st-Succ.↓	Trials/Succ.↓
NADER	22.6%	3.7	4.8
RevoNAS	**47.5%**	**1.4**	**2.3**

Table 6: Quantitative comparison of RevoNAS and baseline NAS methods. The table evaluates performance based on Validity, alongside search efficiency metrics including the number of evaluated designs, GPU hours, and token consumption.

Method	Validity (↑)	# Designs (↓)	GPU Hours (↓)	Tokens (↓)
LLMatic	0.513	2000	0.336	364
RZ-NAS	0.435	1500	0.811	633
NADER	0.422	5-500	0.404	443
RevoNAS	**0.665**	**5**	**0.261**	**281**

improves the positive generation rate by 24.9% and requires only 2.3 attempts on average to obtain a valid positive offspring. It is noteworthy that these results prove that Pareto-based selection is a principled criterion for effective and expandable neural architecture search.

Cost-Effectiveness of Reflective Architecture Search. We further evaluate the overall resource efficiency of RevoNAS against existing baselines, as shown in Tab. 6. A fundamental limitation of prior methods like RZ-NAS, LLMatic or NADER is their reliance on massive design overheads, requiring hundreds to thousands of design evaluations to find a viable architecture. Conversely, RevoNAS achieves a non-trivial structural validity of 0.665 with an efficient set of only 5 designs. By prioritizing high-yield reasoning over brute-force sampling, our framework effectively minimizes resource consumption, achieving notable efficiency with just 0.261 GPU hours and 281 LLM tokens per valid architecture. This highlights RevoNAS as a highly sustainable and efficient solution, offering robust performance with minimal computational and financial overhead.

Applicability Across Search Spaces. To examine whether RevoNAS is applicable beyond conventional convolutional cell-based search spaces, we further evaluate it using a 5M-parameter InternImage-style proxy on ImageNet-1K. As shown in Tab. 7, RevoNAS improves top-1 and top-5 accuracy by 1.20 and 1.55 percentage points, respectively, over the baseline, while reducing ECE from 0.0466 to 0.0364 (all architectures trained for 10 epochs). These simultaneous improvements in classification accuracy and calibration provide evidence that RevoNAS can identify effective architectural refinements beyond the conventional cell-based setting, supporting its applicability to the InternImage-style search space under a limited training budget.

Table 7: Comparison under an InternImage-style 5M proxy setting that preserves the core InternImage block, stem, and downsampling search space.

Methods	Acc.@1 ↑	Acc.@5 ↑	ECE ↓
InternImage-5M	0.4498	0.7012	0.0466
w/ RevoNAS	**0.4618**	**0.7167**	**0.0364**

5 Conclusion

In this work, we proposed RevoNAS, a reflective evolutionary framework for Neural Architecture Search (NAS) that shifts focus from low-level hyperparameter search to global architectural reasoning about high-level structural principles. RevoNAS integrates multi-round multi-expert consensus to accumulate coherent architectural knowledge, adaptive reflective exploration to align exploration and refinement based on reward variance, and Pareto-guided evolutionary selection to ensure both performance and structural diversity. Across CIFAR10/100, ImageNet16-120, COCO-5K, and Cityscapes, RevoNAS consistently achieves superior results while producing architectures that are interpretable and practically deployable. Finally, this finding suggests that neural architecture design benefits significantly from a feedback-aligned iterative reasoning, providing a scalable path toward robust and optimal model search.

AI USE STATEMENT

LLMs are components of the proposed framework and experimental systems, as described in Secs. A, 3 and 4.1. For manuscript preparation, we used AI tools for grammar and style polishing and figure preparation. We did not use AI tools to develop the research ideas, design the experiments, or interpret the results. The authors take full responsibility for all content.

ETHICS STATEMENT

This work studies automated neural architecture search using existing benchmarks and datasets. Improvements in accuracy and computational efficiency do not establish the fairness, privacy, or safety of the resulting models in downstream applications. The search process may also inherit biases from the underlying LLMs and source literature. Deployment requires application-specific evaluation and consideration of potential misuse.

REPRODUCIBILITY STATEMENT

The consensus, reflective exploration, and evolutionary selection mechanisms are described in Secs. 3.2 to 3.4. The experimental protocols, datasets, training settings, LLM configurations, and hyperparameters are documented in Secs. A and 4.1. Additional methodological details are provided in Sec. B, and the multi-agent prompts are presented in Tab. 9. Pseudocode for Pareto-guided selection and multi-round multi-expert consensus is provided in Algorithms 1 and 2, respectively.

REFERENCES

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The journal of machine learning research*, 13(1):281–305, 2012.

Angelica Chen, David Dohan, and David So. Evoprompting: Language models for code-level neural architecture search. *Advances in neural information processing systems*, 36:7787–7817, 2023.

Jizheng Chen, Kounianhua Du, Xinyi Dai, Weiming Zhang, Xihuai Wang, Yasheng Wang, Ruiming Tang, Weinan Zhang, and Yong Yu. Debatecoder: Towards collective intelligence of llms via test case driven llm debate for code generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12055–12065, 2025.

Pei Chen, Boran Han, and Shuai Zhang. Comm: Collaborative multi-agent, multi-reasoning-path prompting for complex problem solving. *arXiv preprint arXiv:2404.17729*, 2024.

Xiangning Chen, Ruochen Wang, Minhao Cheng, Xiaocheng Tang, and Cho-Jui Hsieh. Dr{nas}: Dirichlet neural architecture search. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=9FWas6YbmB3`.

Hyeong Kyu Choi, Xiaojin Zhu, and Yixuan Li. Debate or vote: Which yields better decisions in multi-agent large language models? *arXiv preprint arXiv:2508.17536*, 2025.

Will Dabney, Georg Ostrovski, and André Barreto. Temporally-extended {\epsilon}-greedy exploration. *arXiv preprint arXiv:2006.01782*, 2020.

Anıl Doğru, Mehmet Onur Keskin, and Reyhan Aydoğan. Taking into account opponent's arguments in human-agent negotiations. *ACM Transactions on Interactive Intelligent Systems*, 15(1):1–35, 2025.

Xuanyi Dong and Yi Yang. One-shot neural architecture search via self-evaluated template network. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3681–3690, 2019a.

Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1761–1770, 2019b.

Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*, 2020.

Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *International conference on machine learning*, pp. 1437–1446. PMLR, 2018.

Zhaopeng Feng, Jiayuan Su, Jiamei Zheng, Jiahan Ren, Yan Zhang, Jian Wu, Hongwei Wang, and Zuozhu Liu. M-mad: Multidimensional multi-agent debate for advanced machine translation evaluation. *arXiv preprint arXiv:2412.20127*, 2024.

Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *European conference on computer vision*, pp. 544–560. Springer, 2020.

Kai He, Yucheng Huang, Wenqing Wang, Delong Ran, Dongming Sheng, Junxuan Huang, Qika Lin, Jiaxing Xu, Wenqiang Liu, and Mengling Feng. Crab: A novel configurable role-playing llm with assessing benchmark. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15030–15052, 2025.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Shoukang Hu, Sirui Xie, Hehui Zheng, Chunxiao Liu, Jianping Shi, Xunying Liu, and Dahua Lin. Dsnas: Direct neural architecture search without parameter retraining. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12084–12092, 2020.

Zipeng Ji, Guanghui Zhu, Chunfeng Yuan, and Yihua Huang. Rz-nas: Enhancing llm-guided neural architecture search via reflective zero-cost strategy. In *Forty-second International Conference on Machine Learning*, 2025.

Lars Benedikt Kaesberg, Jonas Becker, Jan Philip Wahle, Terry Ruas, and Bela Gipp. Voting or consensus? decision-making in multi-agent debate. *arXiv preprint arXiv:2502.19130*, 2025.

Yubin Kim, Chanwoo Park, Hyewon Jeong, Yik S Chan, Xuhai Xu, Daniel McDuff, Hyeonhoon Lee, Marzyeh Ghassemi, Cynthia Breazeal, and Hae W Park. Mdagents: An adaptive collaboration of llms for medical decision-making. *Advances in Neural Information Processing Systems*, 37:79410–79452, 2024.

Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *Uncertainty in artificial intelligence*, pp. 367–377. PMLR, 2020.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate. *arXiv preprint arXiv:2305.19118*, 2023.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search, 2019. URL https://arxiv.org/abs/1806.09055.

Yutong Liu, Lida Shi, Rui Song, and Hao Xu. A dual-mind framework for strategic and expressive negotiation agent. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23840–23860, 2025.

Sajad Movahedi, Melika Adabinejad, Ayyoob Imani, Arezou Keshavarz, Mostafa Dehghani, Azadeh Shakery, and Babak N Araabi. Lambda-darts: Mitigating performance collapse by harmonizing operation selection among cells. *arXiv preprint arXiv:2210.07998*, 2022.

Linyong Nan, Ellen Zhang, Weijin Zou, Yilun Zhao, Wenfei Zhou, and Arman Cohan. On evaluating the integration of reasoning and action in llm agents with database question answering. *arXiv preprint arXiv:2311.09721*, 2023.

Muhammad Umair Nasir, Sam Earle, Julian Togelius, Steven James, and Christopher Cleghorn. Llmatic: neural architecture search via large language models and quality diversity optimization. In *proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1110–1118, 2024.

Someen Park, Jaehoon Kim, Seungwan Jin, Sohyun Park, and Kyungsik Han. Predict: multi-agent-based debate simulation for generalized hate speech detection. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 20963–20987, 2024.

Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International conference on machine learning*, pp. 4095–4104. PMLR, 2018.

Md Hafizur Rahman and Prabuddha Chakraborty. Lemo-nade: Multi-parameter neural architecture discovery with llms. *arXiv preprint arXiv:2402.18443*, 2024.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pp. 4780–4789, 2019.

Eduardo Rodrigues Gomes and Ryszard Kowalczyk. Dynamic analysis of multiagent q-learning with ε-greedy exploration. In *Proceedings of the 26th annual international conference on machine learning*, pp. 369–376, 2009.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face, 2023. *arXiv preprint arXiv:2303.17580*, 2023.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

Yuan Tian, Qin Wang, Zhiwu Huang, Wen Li, Dengxin Dai, Minghao Yang, Jun Wang, and Olga Fink. Off-policy reinforcement learning for efficient and effective gan architecture search. In *European conference on computer vision*, pp. 175–192. Springer, 2020.

Michel Tokic. Adaptive ε-greedy exploration in reinforcement learning based on value differences. In *Annual conference on artificial intelligence*, pp. 203–210. Springer, 2010.

Xiaofeng Wang, Zhixin Zhang, Jinguang Zheng, Yiming Ai, and Rui Wang. Debt collection negotiations with large language models: An evaluation system and optimizing decision making with multi-agent. *arXiv preprint arXiv:2502.18228*, 2025.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*, 2023.

Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*, 2018.

Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. *arXiv preprint arXiv:1907.05737*, 2019.

Zekang Yang, Wang Zeng, Sheng Jin, Chen Qian, Ping Luo, and Wentao Liu. Nader: Neural architecture design via multi-agent collaboration. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 4452–4461, 2025.

Peng Ye, Baopu Li, Yikang Li, Tao Chen, Jiayuan Fan, and Wanli Ouyang. b-darts: Beta-decay regularization for differentiable architecture search. In *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10874–10883, 2022.

Yeyong Yu, Runsheng Yu, Haojie Wei, Zhanqiu Zhang, and Quan Qian. Beyond dialogue: A profile-dialogue alignment framework towards general role-playing language model. *arXiv preprint arXiv:2408.10903*, 2024.

Haonan Zhang, Run Luo, Xiong Liu, Yuchuan Wu, Ting-En Lin, Pengpeng Zeng, Qiang Qu, Feiteng Fang, Min Yang, Lianli Gao, et al. Omnicharacter: Towards immersive role-playing agents with seamless speech-language personality interaction. *arXiv preprint arXiv:2505.20277*, 2025a.

Miao Zhang, Steven W Su, Shirui Pan, Xiaojun Chang, Ehsan M Abbasnejad, and Reza Haffari. idarts: Differentiable architecture search with stochastic implicit gradients. In *International Conference on Machine Learning*, pp. 12557–12566. PMLR, 2021.

Pinyi Zhang, Siyu An, Lingfeng Qiao, Yifei Yu, Jingyang Chen, Jie Wang, Di Yin, Xing Sun, and Kai Zhang. Roleplot: A systematic framework for evaluating and enhancing the plot-progression capabilities of role-playing agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12337–12354, 2025b.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024a.

Ruochen Zhao, Wenxuan Zhang, Yew Ken Chia, Weiwen Xu, Deli Zhao, and Lidong Bing. Auto-arena: Automating llm evaluations with agent peer battles and committee discussions. *arXiv preprint arXiv:2405.20267*, 2024b.

Mingkai Zheng, Xiu Su, Shan You, Fei Wang, Chen Qian, Chang Xu, and Samuel Albanie. Can gpt-4 perform neural architecture search? *arXiv preprint arXiv:2304.10970*, 2023.

CONTENTS

A Detailed Formulation and Implementation **1**

A.1 Implementation details 1

A.2 Baseline. 1

A.3 Language model configuration 1

A.4 Datasets 2

A.5 Hyperparameters and Sensitivity (Defaults) 2

B More details of RevoNAD **2**

B.1 Multi-Round Multi-expert Consensus 2

B.2 Adaptive Reflective Design Exploration 5

B.3 Pareto-Guided Evolutionary Selection 5

C Additional Experiments **6**

C.1 Effect of the Underlying LLM 6

C.2 Best-performing models (95.22%) 6

C.3 Structural Design Diversity 7

C.4 Failure Modes 7

A DETAILED FORMULATION AND IMPLEMENTATION

A.1 IMPLEMENTATION DETAILS

All experiments follow the official NAS-Bench-201 training and evaluation settings. Each model is trained for 200 epochs using stochastic gradient descent (momentum 0.9, weight decay 5×10^{-4}) with a cosine-annealed learning-rate schedule starting at 0.1 and a batch size of 256. All experiments are executed on a workstation equipped with four NVIDIA RTX A6000 GPUs.

A.2 BASELINE.

We compare RevoNAD with representative NAS approaches, including random search-based (Bergstra & Bengio, 2012; Li & Talwalkar, 2020), evolutionary, RL-based (Pham et al., 2018; Williams, 1992), HPO-based (Falkner et al., 2018), and differentiable algorithms (Liu et al., 2018; Dong & Yang, 2019a; Hu et al., 2020; Xu et al., 2019; Xie et al., 2018; Zhang et al., 2021; Dong & Yang, 2019b; Chen et al., 2021; Ye et al., 2022; Movahedi et al., 2022; Xu et al., 2019), as well as recent LLM-assisted NAS frameworks (Zheng et al., 2023; Nasir et al., 2024; Ji et al., 2025). We further compare against NAD frameworks such as LeMo-NADe (Rahman & Chakraborty, 2024) and NADER (Yang et al., 2025), which propose architectural designs leveraging LLM-based agents.

A.3 LANGUAGE MODEL CONFIGURATION

All multi-agent components (*Planner*, *Designer*, *Evaluator*) employ GPT-4o as the primary underlying large language model (please refer to Tab. 8). Each agent communicates via structured prompts that encode the current design objectives, architectural constraints, and performance feedback from previously trained models. Unless otherwise specified, we standardize the decoding and context settings across all experiments, using a maximum context length of 4096 tokens, a temperature of 0.7, nucleus sampling with $p = 0.9$, and truncating each response to 512 tokens. In addition to GPT-4o, we also utilize open-source large language models, including Qwen2.5 and DeepSeek, for selected experiments (*i.e.*, cost- or latency-sensitive settings); unless otherwise noted, these models share the same decoding configuration as described above.

Table 8: Notation recap for the RevoNAD framework.

Symbol	Description
$\mathcal{A} = \{a^{(n)}\}$	Architecture search space; set of all candidate architectures.
$a_t^{(n)}$	n-th architecture at generation t.
$\mathcal{I}_t = \{i_t^{(n)}\}$	Inspiration set used at generation t.
E_k	k-th expert LLM agent specialized in axis x_k.
Ω	Consensus module that merges expert proposals into $\mathcal{I}_t$.
s_n	Reflective state at step n (text, rewards, context embeddings).
ε_n	Exploration coefficient at step n in the adaptive ε-greedy policy.
$r_{n-m:n}$	Recent reward window from step $n-m$ to n.
$\pi_n(i \mid s_n)$	Inspiration-sampling policy at step n given state s_n.
$\mathcal{A}_{\text{LLM}}$	LLM-based generator that outputs new inspirations from prompts.
$Q(i)$	Empirical mean reward of inspiration i.
$\sigma(a)$	Performance uncertainty of a estimated from intermediate training dynamics.
$\mathcal{F}_0$	First (non-dominated) Pareto front.
$\{\mathcal{F}_0, \mathcal{F}_1, \dots\}$	Pareto fronts obtained by fast non-dominated sorting of $\mathcal{A}_t$.
$\text{Bid}(a)$	Scalar bidding score combining accuracy, efficiency, structual diversity, confidence, and uncertainty.
$\mathcal{S}_t$	Survivor set at generation t selected for the next evolutionary cycle.

A.4 Datasets

We evaluate RevoNAD on five datasets. The first three follow the NAS-Bench-201 benchmark setting for image classification, while **COCO-5K** and **Cityscapes** are additionally included to examine the scalability and transferability of RevoNAD to object detection and semantic segmentation tasks, respectively. For classification datasets, we apply standard preprocessing with per-channel normalization, random cropping, and horizontal flipping, whereas COCO-5K is processed with per-channel normalization, aspect-ratio–preserving resizing, and random horizontal flipping, following common detection practices. For Cityscapes, we follow standard semantic segmentation preprocessing with per-channel normalization, multi-scale resizing, and random horizontal flipping.

- **CIFAR10:** 60K 32×32 color images from 10 classes. The 50K training images are evenly divided into 25K for training and 25K for validation, while the 10K test images remain unchanged.
- **CIFAR100:** Same resolution as CIFAR10 but with 100 fine-grained categories. The 10K test set is evenly split into 5K for validation and 5K for testing.
- **ImageNet16-120:** A downsampled version of ImageNet16×16, containing 151.7K training, 3K validation, and 3K test images across 120 classes.
- **COCO-5K:** A subset of the COCO dataset, comprising 5K training and 5K validation images with bounding-box and instance-mask annotations across 80 object classes.
- **Cityscapes:** A high-resolution urban scene dataset with 19 evaluation classes. It contains 2,975 training, 500 validation, and 1,525 test images providing fine-grained pixel-level semantic annotations.

A.5 Hyperparameters and Sensitivity (Defaults)

- **MMC:** $\tau_J \in [0.5, 0.8]$, $\tau_Q \in [0.01, 0.05]$, $t_{\min} = 2$, $t_{\max} = 3$.
- **ARDE:** $\varepsilon_{\min} = 0.05$, $\varepsilon_{\max} = 0.8$, $\lambda \in [1, 5]$, window $k = 3 \sim 10$ events.
- **PES:** survival ratio $r = 0.3 \sim 0.6$; $\lambda_p, \lambda_l, \gamma_d, \rho_c, \beta$ chosen so that each normalized term is $\mathcal{O}(1)$ and no single dimension dominates (we use grid-tuned presets).

B More details of RevoNAD

B.1 Multi-Round Multi-expert Consensus

Prompt Design for Multi-Agent Inspiration Extraction. The multi-round consensus framework relies critically on *role-specialized prompting* to ensure that textual architectural reasoning is decomposed, contextualized, and

Table 9: Prompt design for multi-agent architectural ideation.

Prompt Type	Prompt Description
PROMPT_SUBTASKS	**Role:** Senior computer-vision architect. **Goal:** Analyze the paper and extract *core architectural signals* that guide expert agents in decomposing it into design-related sub-tasks. **STRICT JSON output:** • `"tasks"`: [main CV tasks (e.g., detection, segmentation, 3D reconstruction, OCR, SLAM, BEV)] • `"sub_tasks"`: [$\leq$4 architecture-influential axes (e.g., hierarchy design, receptive-field control, routing/sparsity, normalization layout, parameter sharing)] • `"keywords"`: [$\leq$5 ops or motifs (e.g., deformable conv, attention window, SE block, MoE router, adapter, residual gating)] **Guidelines:** Focus on architectural levers (hierarchy, connectivity, mixing, routing, memory, inductive bias, interface, stability); avoid dataset or training details; keep concise and transferable.
PROMPT_EXPERT	**Role:** Expert agent for a specific sub-task (`"field"`). **Goal:** Propose one concise, local, drop-in architectural improvement related to the given field, applicable across CV pipeline stages (e.g., encoding, multi-scale aggregation, temporal/geometric reasoning, cross-modal integration, output mapping). **Context:** • `PAPER (numbered)`: paper_numbered • `CURRENT INSPIRATIONS`: current_insp **STRICT JSON output:** • `"proposal"`: $\leq$40-word architectural recipe (e.g., DWConv$\rightarrow$SE$\rightarrow$MLP, deformable attention with shift) • `"rationale"`: $\leq$40 words explaining performance/efficiency/generalization benefit • `"evidence_refs"`: [supporting sentence indices] **Guidelines:** Be component-agnostic; focus on architecture-level reasoning; refine or merge if similar ideas exist, otherwise propose a novel one.
PROMPT_MERGE	**Role:** Orchestrator agent integrating expert proposals. **Goal:** Merge, deduplicate, and refine expert JSON proposals into a concise, diverse shortlist of architectural inspirations. **Input:** Expert JSON proposals (array of JSON objects). **STRICT JSON output:** • `"inspirations"`: [final concise architectural ideas ($\leq$40 words each, up to 4 items)] **Guidelines:** Merge compatible ideas succinctly; remove redundancy; ensure diversity across scale, efficiency, temporal, and modality reasoning; keep each item modular and transferable.

refined in a structured way. Rather than prompting a single LLM instance to extract architectural ideas in one pass, we define three complementary prompt types corresponding to the three core roles in MMC: (1) sub-task decomposition, (2) expert proposal, and (3) proposal integration. Here, Tab. 9 summarizes these prompts. We highlight several key design principles:

- **Decomposition before Ideation:** PROMPT_SUBTASKS forces the LLM to first identify *architecture-influential axes* (*e.g.*, hierarchy depth, spatial aggregation strategy, skip-connectivity, feature routing), preventing single-perspective inspiration collapse.
- **Locality and Modularity:** PROMPT_EXPERT constrains each expert to output a *single, concise, drop-in architectural cue* with explicit performance rationale. This prevents verbose conceptual drift

Algorithm 2: Multi-Round Multi-expert Consensus

Input:
Paper P; experts $\{E_1, \ldots, E_K\}$; thresholds τ_J, τ_Q; round limits $t_{\min}, t_{\max}$;
Output:
Stable, diverse inspiration set $\mathcal{I}$

$I^{(0)} \leftarrow \emptyset$, $t \leftarrow 0$
while *not converged and* $t < t_{\max}$ **do**
  foreach E_k *in experts (in parallel)* **do**
    $p_k^{(t)} \leftarrow E_k(P, \mathcal{I}^{(t-1)})$
  end
  $\mathcal{I}^{(t)} \leftarrow \Omega(\{p_k^{(t)}\}_{k=1}^K)$
  `# Consensus stability based on set overlap`
  $J \leftarrow \frac{|\mathcal{I}^{(t)} \cap \mathcal{I}^{(t-1)}|}{|\mathcal{I}^{(t)} \cup \mathcal{I}^{(t-1)}|}$
  `# Mean quality score based on reflection rewards; r(i) is the reflection reward of inspiration i`
  $\mu^{(t)} \leftarrow \frac{1}{|\mathcal{I}^{(t)}|} \sum_{i \in \mathcal{I}^{(t)}} r(i)$
  $\Delta_\mu \leftarrow \frac{|\mu^{(t)} - \mu^{(t-1)}|}{|\mu^{(t-1)}| + \epsilon}$
  if $J \geq \tau_J$ ***and*** $\Delta_\mu \leq \tau_Q$ ***and*** $t \geq t_{\min}$ **then**
    converged $\leftarrow$ True
  end
  $t \leftarrow t + 1$
end
return $\mathcal{I}^{(t-1)}$

Table 10: Stability of sub-axis labels across multiple random seeds. Higher consistency and lower variability indicate a more reliable decomposition.

Metric	Value	Interpretation
Sub-axis Consistency (%)	**84.98**	Most sub-axis labels recur across runs
Standard Deviation (%)	**16.59**	Moderate variability for a subset of axes

and ensures each output corresponds to a manipulable graph-level transformation (consistent with $\mathcal{T}(a, i)$ in the main method).

- **Redundancy Control via Structured Merging:** PROMPT_MERGE enforces JSON-based filtering and clustering, ensuring the orchestrator $\mathcal{M}$ merges proposals into a *minimal*, *diverse*, and *non-overlapping* inspiration set.

Expert Decomposition. Given a set of authentic papers P with K conceptual axes (*e.g.*, hierarchy, receptive-field control, skip topology, gating/attention, normalization/activation, regularization, token routing), we instantiate experts $\{E_k\}_{k=1}^K$. Each expert specializes in one axis and proposes axis-specific cues conditioned on the current shared set $\mathcal{I}^{(t-1)}$:

$$p_k^{(t)} = E_k\big(P, \mathcal{I}^{(t-1)}\big), \tag{5}$$

yielding a multiset of proposals $\{p_k^{(t)}\}_{k=1}^K$. Proposals are short, "drop-in" primitives that can be composed in the block-graph grammar.

Consensus Operator. We use an orchestrator Ω to merge, filter, and deduplicate proposals:

$$\mathcal{I}^{(t)} = \Omega\big(\{p_k^{(t)}\}_{k=1}^K\big). \tag{6}$$

Ω performs lexical and semantic consolidation of proposals (implemented via an LLM merge prompt), while our outer loop uses sentence embeddings for stability and diversity checks.

Stopping Criterion and Stability. We check both set-stability and quality-plateau:

$$J\big(\mathcal{I}^{(t)}, \mathcal{I}^{(t-1)}\big) \geq \tau_J, \quad \Delta_\mu^{(t)} \triangleq \frac{|\mu^{(t)} - \mu^{(t-1)}|}{|\mu^{(t-1)}| + \epsilon} \leq \tau_Q, \tag{7}$$

with a minimum round count $t_{\min}$ to avoid premature stopping. This is exactly reflected in Alg. 2 of the main paper.

Reliability of Sub-axis Decomposition. To assess the reliability of our MMC-based sub-task decomposition, we repeatedly run the sub-task extractor under five independent seeds, as shown in Tab. 10. For each paper, we collect the set of sub-axis labels (`sub_tasks`) produced in each run, and then treat each distinct label as a sub-axis candidate. We focus on *recurring* sub-axes that appear in at least two runs, and measure their stability as

$$\text{consistency}(\ell) \;=\; \frac{\#\{\text{runs where label } \ell \text{ appears}\}}{\#\{\text{runs for that paper}\}} \in [0,1]. \tag{8}$$

As a result, we obtain an average consistency of 84.98% with a standard deviation of 16.59%. This indicates that, once a sub-axis emerges as a recurring design dimension, it is typically recovered in the vast majority of runs, with only moderate variability. In other words, most architectural axes identified by our system are not artifacts of a particular prompt realization, but persist as stable roles under independent re-samplings of the LLM. The remaining variance reflects a smaller subset of more ambiguous or task-specific axes that are naturally sensitive to prompting and seed choices, highlighting both the robustness and residual stochasticity of the decomposition process.

Complexity. Each round costs $O(K + |\mathcal{I}^{(t)}|^2)$ in the worst case due to pairwise similarity computations used for stability and diversity monitoring. In practice, we cap the number of proposals per expert, so $|\mathcal{I}^{(t)}|$ remains small and the overhead is negligible in our experiments.

B.2 Adaptive Reflective Design Exploration

Redundancy Filter. Before materialization, we apply redundancy pruning over proposals using semantic radius δ in embedding space:

$$\text{Keep } i \text{ only if } \min_{j \in \text{Kept}} d_{\cos}\big(\phi(i), \phi(j)\big) \geq \delta.$$

We also drop proposals that violate hard feasibility checks derived from block-graph templates (e.g., incompatible channel sizes or stride cascades).

Exploration/Exploitation Gate. ARDE (Sec. 3.2) modulates the fraction of *exploratory* proposals admitted. When reward variance falls, we accept a larger share of structurally novel or out-of-axis proposals; when variance spikes, we prioritize exploitative refinements around high-utility inspirations.

B.3 Pareto-Guided Evolutionary Selection

Structural Diversity. We require a structural diversity signal between a child block G_c and its parent G_p that is (i) cheap to compute for hundreds of candidates and (ii) more stable than exact graph edit distance (GED), which is sensitive to small local rewires. To this end, we define a bounded structural similarity $S \in [0,1]$ and use

$$\text{StructDiv.}(G_c, G_p) \;=\; 1 - S, \tag{9}$$

so that higher values indicate more structurally novel children. Specifically, each block is represented as a directed graph $G = (V, E)$ with node labels $\ell_G(v)$ denoting operation types. We combine three complementary similarities: (1) Operation composition. We first compare the multiset of operation types via a Jaccard similarity

$$h_G(o) = \big|\{v \in V_G \mid \ell_G(v) = o\}\big|, \tag{10}$$

$$s_{\text{op}} = \frac{\sum_o \min\big(h_c(o), h_p(o)\big)}{\sum_o \max\big(h_c(o), h_p(o)\big)}, \tag{11}$$

with the convention $s_{\text{op}} = 1$ if the denominator is zero. This term checks whether parent and child use a similar mix of building blocks. (2) Depth-aware placement. Then, we assign each node a topological depth $d_G(v)$ and build, for each operation o, a depth histogram $c_G^{(o)}(k)$ and normalized distribution $p_G^{(o)}(k)$. For each o, we compute a 1D Wasserstein-1 distance along the depth axis via cumulative distributions and convert it to a similarity $s_{\text{depth}}^{(o)} = 1 - \text{EMD}^{(o)}$. Averaging over all operations that appear in either graph gives

$$s_{\text{depth}} = \frac{1}{|\mathcal{O}'|} \sum_{o \in \mathcal{O}'} s_{\text{depth}}^{(o)}, \tag{12}$$

which captures whether the same operations are placed at similar stages of the computation. (3) Local wiring motifs. Finally, we run a few iterations (T=2) of Weisfeiler–Lehman color refinement, starting from operation labels, and build a color histogram h_G^{WL} at the final iteration. We then compute the cosine similarity

$$s_{\text{WL}} \;=\; \frac{\sum_\gamma h_c^{\text{WL}}(\gamma) h_p^{\text{WL}}(\gamma)}{\sqrt{\sum_\gamma h_c^{\text{WL}}(\gamma)^2}\,\sqrt{\sum_\gamma h_p^{\text{WL}}(\gamma)^2}}, \tag{13}$$

Table 11: Effect of the underlying LLM on our framework.

Model	CIFAR10 (↑)		CIFAR100 (↑)		ImageNet16-120 (↑)	
	Val.	Test	Val.	Test	Val.	Test
DeepSeek	91.28	94.49	75.18	74.83	48.87	49.13
Qwen2.5	91.17	94.77	76.21	76.25	**51.00**	50.55
GPT-4o	**92.55**	**95.22**	**76.70**	**76.38**	50.58	**50.72**

which provides a lightweight proxy for higher-order wiring patterns. Here, the three terms are combined as

$$S_{\text{raw}} = 0.50\, s_{\text{op}} + 0.30\, s_{\text{depth}} + 0.20\, s_{\text{WL}}, \tag{14}$$

$$S = \min\big(1,\ \max(0, S_{\text{raw}})\big), \tag{15}$$

and

$$\text{StructDiv.}(G_c, G_p) = \begin{cases} 1, & \text{if } G_c \text{ or } G_p \text{ is missing,} \\ 1 - S, & \text{otherwise.} \end{cases} \tag{16}$$

This factorized design is significantly cheaper than full GED, yet still aligns with architectural intuition: novel children should differ in which operations they use, where they place them, and how they wire them into local motifs.

Confidence. Given a trained classifier f_θ and an evaluation set $\mathcal{D} = \{x_i\}_{i=1}^N$, we define the predictive probability vector for each input as

$$\mathbf{p}_i \ = \ \text{softmax}\big(f_\theta(x_i)\big) \in \mathbb{R}^C, \tag{17}$$

where C is the number of classes. The per-sample confidence is the maximum class probability

$$c_i \ = \ \max_{1 \le k \le C} p_{i,k}. \tag{18}$$

The reported *robustness* score corresponds exactly to the average confidence over all test samples:

$$\text{Conf.} \ = \ \frac{1}{N}\sum_{i=1}^{N} c_i \ = \ \frac{1}{N}\sum_{i=1}^{N} \max_k \text{softmax}\big(f_\theta(x_i)\big)_k. \tag{19}$$

C Additional Experiments

C.1 Effect of the Underlying LLM

Tab. 11 reports the impact of different underlying LLMs on our framework. On CIFAR10, GPT-4o achieves the best validation and test accuracies (92.55% / 95.22%), outperforming DeepSeek (91.28% / 94.49%) and Qwen2.5 (91.17% / 94.77%). On CIFAR100, the differences between GPT-4o and Qwen2.5 become smaller: GPT-4o reaches 76.70% / 76.38% (val/test), while Qwen2.5 attains 76.21% / 76.25%. On the more challenging ImageNet16-120 benchmark, performance across LLMs is largely comparable. Qwen2.5 obtains the highest validation accuracy (51.00%), whereas GPT-4o delivers the best test accuracy (50.72%), with all gaps within 0.6 points. These results indicate that our framework is not overly sensitive to the particular LLM instantiation: stronger models such as GPT-4o provide consistent but moderate gains, while more lightweight alternatives (DeepSeek, Qwen2.5) still produce competitive architectures without collapsing the search.

C.2 Best-performing models (95.22%)

As shown in Tab. 12, the reflective explorer effectively improves the legacy ResNet basic block, yielding non-trivial performance accuracy (up to 95.22%). The discovered key modifications can be categorized into three major themes. First, Dynamic Routing and Sparsity: The highest-performing variant, p1483 (95.22%), introduces a per-block top-k expert router (MoE-lite), indicating that dynamic capacity allocation is highly effective even at the micro-block level. Similarly, p4199 validates the benefit of cross-token routing. Second, Structural Modernization: Inspired by modern architectures like ConvNeXt and Transformers, p965 (95.12%) successfully transitions standard convolutions into a Depthwise-FFN (DW-FFN) design. Third, Normalization and Feature Stability: Variants like p928 consistently favors pre-normalization strategies (*e.g.*, RMSNorm swap-in) over traditional post-activation BN, while p1865 and p856 improve spatial stability through cross-scale fusion and anti-aliasing. Together, these inspirations confirm that RevoNAD can autonomously discover highly sophisticated, informative design principles from a simple baseline.

Table 12: Architectures generated by the auto-reflective ε-greedy explorer starting from a ResNet basic block (CIFAR10). We list the strongest and diverse variants (omitting degenerate 0/10% runs).

Inspiration ID	Inspiration	Acc. (%)	Key Architectural Modification
p1483	`Per-block top-`k `expert router (MoE-lite) with load-balance loss`	95.22	DW-FFN base; add block-wise routing/gating idea (MoE-lite) while keeping residual PW mixing.
p965	`DW-FFN: depthwise conv inside MLP for token mixing`	95.12	Compact DWConv(3 × 3)→PWConv; remove last ReLU; pre-activation-style add.
p1865	`In-block cross-scale fusion (stage` $k \leftrightarrow k\pm1$`)`	94.69	Extra 1×1 projection branch merged into residual to blend cross-scale features.
p4199	`Cross-token routing head (top-`k `sparse links)`	94.61	Add light routing on DW-FFN output path to sparsify cross-token flow.
p928	`Use pre-norm residual connections for block stability`	94.57	BN→Conv→ReLU pre-activation; residual add before final BN.
p856	`Anti-alias downsample branch (blur∼dilation)`	94.17	Replace stride-2 with dilation on 2nd conv to suppress aliasing in residual path.

Table 13: Transfer studies on (a) COCO-5K and (b) Cityscapes.

(a) Object Detection & Inst. Seg.

Method	COCO-5K		
	AP_{bbox} (↑)	$AP@50_{bbox}$ (↑)	AP_{mask} (↑)
YOLOv8	17.2	27.7	18.5
w/ RevoNAD	**18.0**	**28.1**	**18.9**

(b) Semantic Segmentation.

Method	Cityscapes		
	aAcc (↑)	mAcc (↑)	mIoU (↑)
SemanticFPN	95.83	81.80	74.01
w/ RevoNAD	**95.88**	**82.73**	**75.05**

C.3 Structural Design Diversity

We study how the structural-diversity weight γ_d in our PES influences the discovered blocks. For each dataset (CIFAR10, CIFAR100, and ImageNet16-120), we run RevoNAD with $\gamma_d \in \{0.1, 0.3\}$ while keeping all other hyperparameters fixed and visualize the resulting best blocks. On CIFAR10 Fig. 6, increasing the diversity weight from 0.1 to 0.3 produces a slightly richer topology with additional residual paths and aggregation operations, which translates into a modest but consistent accuracy gain (from 94.02% to 94.43%). The effect is more pronounced on CIFAR100 Fig. 7. With $\gamma_d = 0.3$, the discovered block exhibits deeper stacking and more heterogeneous branches compared to the relatively shallow pattern obtained with $\gamma_d = 0.1$, leading to a substantial improvement from 72.05% to 75.84% accuracy. In contrast, on ImageNet16-120 Fig. 8, a stronger emphasis on diversity slightly degrades performance: the model found with $\gamma_d = 0.3$ is structurally more fragmented, but its accuracy drops from 50.42% to 48.72%. Overall, it is noteworthy that a moderate increase in γ_d yields clear gains on CIFAR10/100 by promoting new architectural motifs, whereas on ImageNet16-120 an overly strong diversity prior can harm performance, suggesting that γ_d should be tuned to balance novelty and reliability.

C.4 Failure Modes

Inspiration Collapse. As shown in Tab. 15, RevoNAD occasionally produces infeasible or non-viable architectures for two main reasons. First, although our MMC module generates structurally meaningful inspirations via iterative collaborative discussions, the underlying LLM can still hallucinate structural design motifs. Second, even though the inspiration memory is populated with semantically diverse design cues, a relatively high exploration rate ε_n can steer the search toward mismatched or non-informative architectural clues, which are more likely to lead to invalid or degenerate architectures. Due to these limitations, bridging architectural search with LLM-driven generation remains a challenging problem.

Developing failures. NAD effectively addresses the narrow, hand-crafted search spaces of conventional NAS and can propose well-informed designs grounded in high-level architectural knowledge. However, NAD still faces difficulties in consistently aligning high-level specifications with low-level operator modules, mainly due to detailed code implementation error. As illustrated by the error logs in Table 16, the generated blocks suffer from

Table 14: Prompt-variant evaluation from the same parent model.

Prompt	Prompt Text	Val.	Test
Parent	–	90.80	94.10
Original (P_{orig})	`Input-conditioned channel gating for feature enhancement`	91.31	94.75
Syntactic (P_{syn})	`Use input features to generate channel-wise gates for representation enhancement`	91.17	94.47
Tone (P_{tone})	`Introduce a lightweight input-aware gating module for adaptive feature refinement`	91.38	94.82
Lexical (P_{lex})	`Input-adaptive channel reweighting for feature refinement`	91.26	94.49

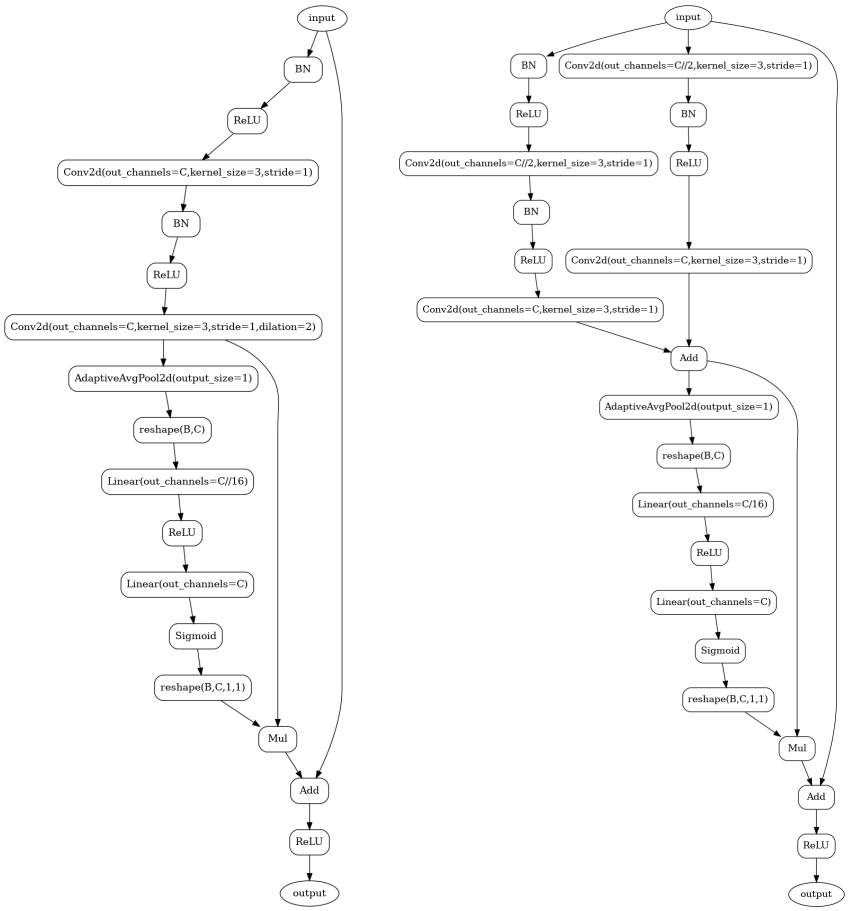
Table 15: Examples of inspiration collapse across datasets.

Dataset	Acc. (%)	Inspiration
CIFAR10	0.0	reprojection-based feature alignment
CIFAR10	0.0	Design with passive solar orientation to enhance energy efficiency and comfort.
CIFAR100	1.0	in-block cross-scale fusion (stage k2194 ↔ k00b11) via 1x1 proj + add
ImageNet16-120	0.83	Adaptive reuse of historic buildings with modular interiors for modern functionality and sustainability.
ImageNet16-120	6.37	Biophilic skyscrapers integrating vertical gardens for sustainable urban living.
ImageNet16-120	13.43	Solar-powered community centers with adaptive spaces for diverse cultural events.

implementation-time failures such as invalid output shapes, mismatched batch or channel dimensions, illegal multi-input usage of operators (e.g., `Conv2d` or `repeat`), undefined operations (*e.g.*, `max_pool2d`), unused or multiply defined nodes, and computation graphs that are not directed acyclic graphs. In this section, we present a practical limitation of our pipeline: a non-trivial portion of the search budget is spent on architectures that appear syntactically plausible at the design level but cannot be realized as valid, trainable models.

Table 16: Failure cases during architecture developing.

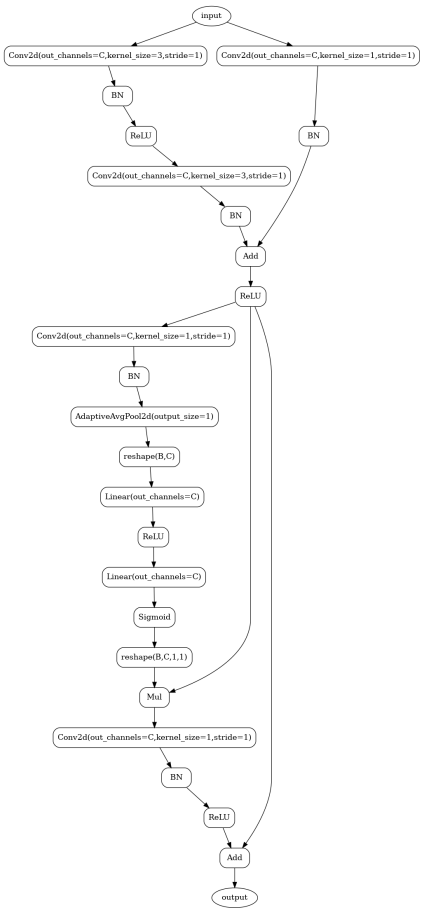
Error logs
node 1 error: Output shape must be (B,dim,H,W).
node 23 error: reshape operation's first dimension must be B.
node 21 error: The node is not used.
node 7 error: The node is not used.
node 19 error: Add operation's inputs can not be Added.
node 8 error: Undefined computation max_pool2d is used.
node 10 error: When using Linear, the last dimension must be the channel dimension.
node 6 error: Conv2d operation can receive only one input.
node node error: H and W cannot appear in layer definition.
shape '[4, 256]' is invalid for input of size 2048.
node 1 error: The height and width of the output feature map must be 1.0 of the input feature map.
node 19 error: repeat operation can receive only one input.
output must be the only output node.
The computation graph of the block is not a directed acyclic graph.

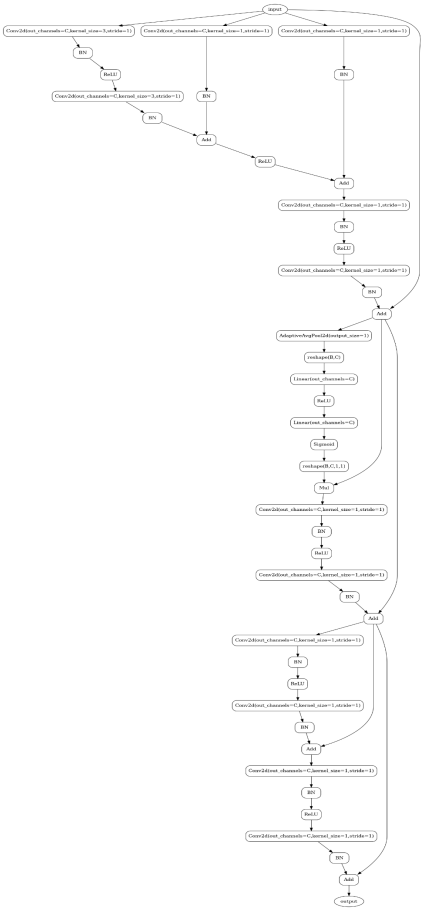


Structural diversity : 0.1
Accuracy :0.9402

Structural diversity: 0.3
Accuracy: 0.9443

Figure 6: Structural Diversity on CIFAR10

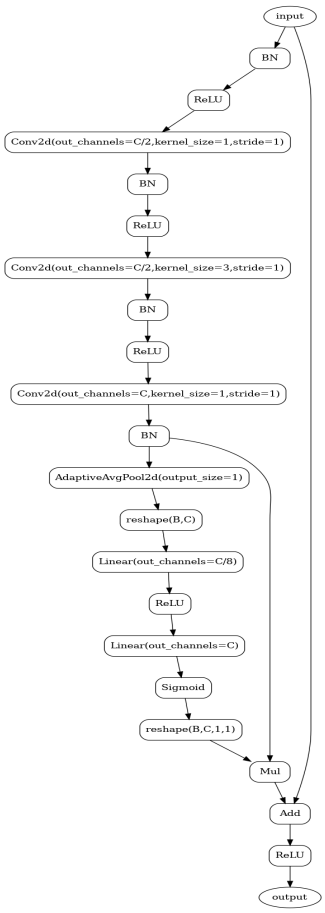




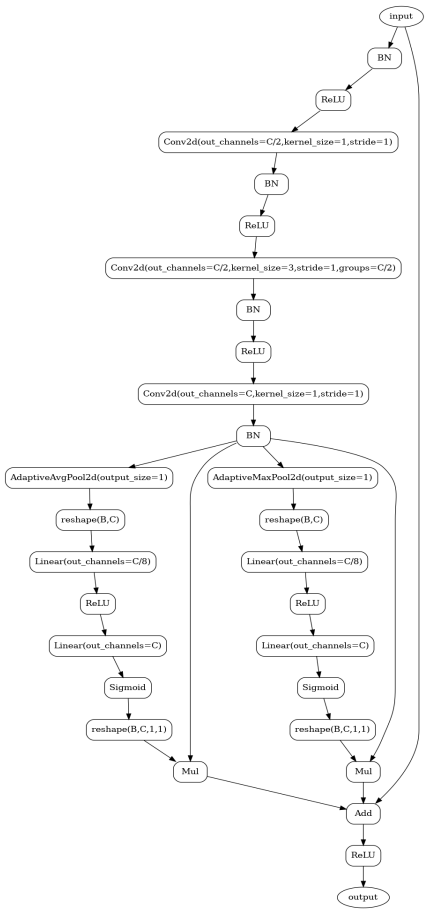
Structural diversity: 0.1
Accuracy: 0.7205

Structural diversity: 0.3
Accuracy: 0.7584

Figure 7: Structural Diversity on CIFAR100



Structural diversity: 0.1
Accuracy: 0.5042



Structural diversity: 0.3
Accuracy: 0.4872

Figure 8: Structural Diversity on ImageNet16-120