LENS: Learning Architecture Navigator for LLM Agentic Systems

Guancheng Wan[*], **Jiayi Yang**[*], **Mengting Li**[*]
University of California, Los Angeles

Abstract

Large Language Model (LLM)-empowered multi-agent systems extend the cognitive boundaries of individual agents through disciplined collaboration, while constructing these systems often requires labor-intensive manual designs. A frontier effort to automate this process is to optimize an Agentic Supernet, a probabilistic distribution of architectures from which query-dependent workflows can be dynamically sampled. However, while this paradigm allows for dynamic resource allocation, its underlying optimization process presents a critical performance bottleneck: inconsistent architectural feedback suppresses reliable credit assignment and prematurely narrows exploration, missing innovative and efficient designs. To address this, we introduce **LENS** (**L**earning-**E**nhanced **N**eural **S**earch for Agentic Workflows), a dual-module framework that systematically resolves both challenges. The *Adaptive Diversity Module (ADM)* maintains comprehensive exploration across the architectural space, while the *Retrospective Guidance Module (RGM)* learns from historical evaluations to provide stable search direction. By decoupling diversity maintenance from directional guidance, LENS achieves robust search that discovers higher-utility, lower-cost configurations. Comprehensive evaluations across diverse benchmarks demonstrate that LENS is: **(I) higher-performing**, achieving up to 13.63% accuracy improvement on challenging benchmarks with the same search budget; **(II) more sample-efficient**, requiring only 30 training samples to outperform baselines trained on much larger datasets; and **(III) more cost-effective**, reducing inference token consumption by 7.8% while significantly improving performance.

Introduction

Recent advancements in Large Language Models (LLMs) have demonstrated remarkable capabilities in natural language understanding, reasoning, and generation (Brown et al. 2020). This progress has catalyzed the development of LLM-powered agents, which can autonomously perform complex tasks by leveraging tools, planning, and interacting with their environment (Wei et al. 2023). These agents are increasingly organized into Multi-Agent Systems (MAS), where multiple agents collaborate to solve problems that are beyond the capabilities of a single agent (Russell and Norvig 2020). Frameworks such as AutoGen (Wu et al. 2024) and CAMEL (Li et al. 2023) have emerged to facilitate the construction of these collaborative systems. However, the de-

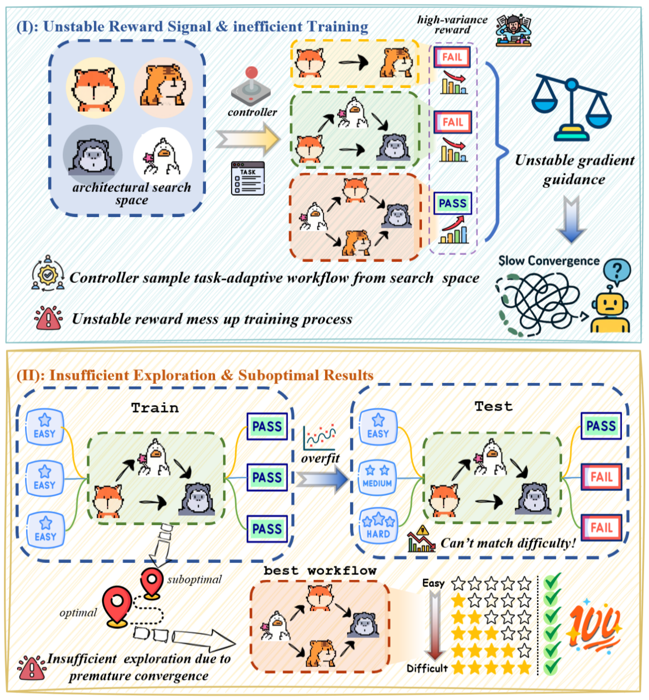

(I): Unstable Reward Signal & inefficient Training
high-variance reward
architectural search space
controller
FAIL
FAIL
PASS
Unstable gradient guidance
Slow Convergence
Controller sample task-adaptive workflow from search space
Unstable reward mess up training process
(II): Insufficient Exploration & Suboptimal Results
Train
EASY
EASY
EASY
PASS
PASS
PASS
overfit
Test
EASY
MEDIUM
HARD
PASS
FAIL
FAIL
Can't match difficulty!
suboptimal
optimal
best workflow
Easy
Difficult
Insufficient exploration due to premature convergence

Figure 1: Illustration of the two core optimization challenges in the Agentic Supernet paradigm. **(I) Unstable Guidance:** The controller, guided by inconsistent architectural feedback from the environment, follows a highly erratic and inefficient search trajectory, struggling to identify a clear and consistent direction for improvement. **(II) Insufficient Exploration:** This inherent instability, coupled with a lack of explicit exploration incentives, often causes the search to prematurely converge into the "gravity well" of a suboptimal architecture, failing to discover potentially more innovative and superior solutions in other regions of the vast search space. Our `MASTER` framework is designed to effectively resolve these two specific issues.

[*]These authors contributed equally.

sign of these systems has largely been a manual and static process, limiting their adaptability and posing a significant challenge in fully harnessing the collective intelligence of agents (Zhang et al. 2025b).

To overcome this, the research focus has rapidly shifted towards the automation of agentic workflows. This field has progressed from optimizing individual components (Khattab et al. 2023; Zhuge et al. 2024) to end-to-end architecture search (Hu, Lu, and Clune 2025; Shang et al. 2025; Zhang et al. 2025c). Against this backdrop, `MaAS` (Zhang et al. 2025b) introduced a groundbreaking paradigm: the Agentic Supernet. Instead of searching for a single, static optimal solution, it optimizes a probability distribution over a vast space of candidate architectures. Using a controller, `MaAS` can dynamically sample a tailored multi-agent architecture based on the specifics of an input query, achieving a better balance between problem-solving and resource allocation.

However, while the Agentic Supernet paradigm allows for dynamic workflow generation, its underlying optimization process still presents a fundamental performance bottleneck. The inherent instability of the architectural feedback used for learning curtails effective exploration and leads to inefficient training. This observation raises a critical question: **I)** ***How can we effectively stabilize the architecture search process against inconsistent architectural feedback?*** Addressing the inconsistency in architectural feedback is the first crucial step towards a robust search algorithm. However, a stable learning process alone does not guarantee the discovery of globally optimal architectures. The controller, now guided by more reliable feedback, might become highly efficient at local optimization, quickly converging to the first plausible architecture it discovers. This behavior arises because the search space is vast and complex, containing many "good-enough" solutions that act as local minima. Without an explicit incentive to explore, the controller has no reason to deviate from these safe, familiar patterns to investigate more novel, potentially higher-reward architectural configurations that may lie in distant regions of the search space. This tendency towards conservative improvement raises another question: **II)** ***How to ensure comprehensive exploration to prevent premature convergence to suboptimal architectures?***

To simultaneously address the challenges mentioned above, we propose **LENS** (**L**earning-**E**nhanced **N**eural **S**earch for Agentic Workflows), a dual-module framework designed for robust architecture search. To answer question **I**), LENS incorporates the *Retrospective Guidance Module (RGM)* that learns from historical evaluations to provide stable search direction, replacing noisy feedback with reliable guidance. To address question **II**), LENS features the *Adaptive Diversity Module (ADM)* that systematically maintains exploration breadth throughout the search process, preventing premature convergence to local optima. By decoupling these two critical functions—stable guidance and comprehensive exploration—LENS achieves a robust search process that discovers innovative and globally optimal architectures. Our principal contributions are summarized as follows:

❶ ***Problem Formulation.*** We are the first to systematically identify and formulate the core optimization challenges of the Agentic Supernet paradigm as two distinct problems: inconsistent search direction and insufficient exploration breadth.

❷ ***Practical Solution.*** We propose `LENS`, a dual-module framework featuring the Retrospective Guidance Module (RGM) for stable directional guidance and the Adaptive Diversity Module (ADM) for comprehensive exploration, achieving robust architecture search without relying on noisy feedback.

❸ ***Experimental Validation.*** We conduct comprehensive experiments on multiple benchmarks, demonstrating that `LENS` significantly outperforms existing automated design methods in terms of convergence speed, final architecture quality, and cost-effectiveness.

Related Work

LLM Agents and Agentic Systems

The adventage of Large Language Models (LLMs) has given rise to powerful autonomous agents (Shen et al. 2023; Zhu et al. 2024). A single agent, by integrating external tools, memory modules, and planning capabilities, can already handle complex tasks. However, the capabilities of a single agent are ultimately limited. Research has shown that organizing multiple agents into a Multi-agent System (MAS) can effectively raise the ceiling on problem-solving capabilities through well-designed interaction and collaboration mechanisms (Wang et al. 2024). Early representative works, such as AutoGen (Wu et al. 2024), LLM-Debate (Du et al. 2024), and AgentVerse (Chen et al. 2023b), explored different collaborative models like role-playing and multi-round debates, validating the immense potential of multi-agent collaboration. However, these systems typically require tedious manual configuration by domain experts, which limits their generality and scalability.

Automation of Agentic Workflows

To reduce manual effort and enhance system adaptability, automating the construction of agentic workflows has become a research hotspot. Research in this area can be broadly categorized into: **(I) Prompt Optimization**, where methods like PromptBreeder (Fernando et al. 2023) and DsPy (Khattab et al. 2023) use algorithms to automatically generate or refine prompts for specific tasks; **(II) Inter-Agent Communication Optimization**, where frameworks like GPTSwarm (Zhuge et al. 2024) and DyLAN (Liu et al. 2024) focus on optimizing the interaction structure or topology among agents; and **(III) Agent Role/Profile Evolution**, where approaches like EvoAgent (Yuan et al. 2024) and AutoAgents (Chen et al. 2023a) use evolutionary algorithms to automatically generate agent profiles.

More recently, research has moved towards more comprehensive, end-to-end automation. ADAS (Hu, Lu, and Clune 2025), AgentSquare (Shang et al. 2025), and AFlow (Zhang et al. 2025c) can automatically construct entire workflows within a vast design space using heuristic search, evolutionary algorithms, or Monte Carlo Tree Search (MCTS).

EvoFlow (Zhang et al. 2025a) further models this as a multi-objective optimization problem, aiming to find a population of workflows that balance cost and performance. Our direct baseline, `MaAS` (Zhang et al. 2025b), revolutionizes this by introducing the **Agentic Supernet**, shifting the search target from a fixed workflow to an architectural distribution, enabling the dynamic generation of workflows based on queries. The work in this paper builds upon the `MaAS` paradigm, focusing specifically on resolving its core optimization algorithm's limitations.

Optimization Methods in Automated Design

The process of automated agent system design shares a strong resemblance with **Neural Architecture Search (NAS)** in the broader machine learning field (Ren et al. 2021). Many classic NAS techniques, such as reinforcement learning (RL), evolutionary algorithms, and Bayesian optimization, have been successfully applied to the automated search for agentic workflows.

In particular, the pioneering work in NAS by Zoph et al. (Zoph and Le 2017) utilized the **REINFORCE** policy-based learning algorithm to train a controller for generating network architectures. However, as subsequent research in the NAS field revealed, REINFORCE was gradually superseded by more efficient methods due to its unstable feedback problem. The `MaAS` framework faces this very same challenge when optimizing its Agentic Supernet. This paper draws inspiration from the mature ideas that have evolved within the field of automated design to solve inconsistent optimization feedback problems. By introducing a learned value function as a stabilization baseline, we provide stable and reliable guidance for the `MaAS` controller's exploration in its vast search space. This measure is not merely the application of a new RL algorithm but a targeted and necessary technical upgrade to solve the inherent optimization difficulties of the `MaAS` paradigm.

Method

To address the core challenges of inconsistent search direction and insufficient exploration in the Agentic Supernet paradigm, we propose the LENS framework. LENS is built upon a dual-module architecture that decouples two critical functions: (1) maintaining comprehensive exploration across the vast architectural space, and (2) providing stable directional guidance derived from historical experience. This section comprehensively details the design and operation of the two core modules.

Adaptive Diversity Module (ADM)

The ADM is responsible for maintaining comprehensive exploration throughout the search process. Its design addresses the fundamental challenge that architectural search spaces are vast and complex, containing numerous local optima that can trap naive search strategies.

Architecture and Function. The ADM operates through a **Sampler** network π_ϕ that generates candidate multi-agent workflows. Given a query and the current search context (represented as state s_t), the Sampler outputs a probability distribution $\pi_\phi(a_t|s_t)$ over the available agentic operators $\mathbb{O}$. At each step t, an operator a_t is selected according to this distribution, and this process continues until a complete workflow $\mathcal{G} = (a_0, a_1, \dots, a_{L-1})$ is constructed.

The key innovation of ADM lies not merely in sampling, but in its systematic diversity maintenance mechanism. Rather than allowing the Sampler to converge quickly to a narrow set of familiar patterns, ADM actively monitors and regulates the distribution breadth. This is achieved through a diversity metric that quantifies the spread of the sampling distribution:

$$\begin{aligned} D(\pi_\phi) &= \sum_{t=0}^{L-1} D_t(\pi_\phi(\cdot|s_t)) \\ &= -\sum_{t=0}^{L-1} \sum_{a\in\mathbb{O}} \pi_\phi(a|s_t) \log \pi_\phi(a|s_t). \end{aligned} \tag{1}$$

By incorporating this diversity metric into the Sampler's objective, ADM ensures that the search maintains a healthy balance between exploiting promising architectural patterns and exploring novel combinations. This mechanism prevents the premature collapse of the search into local optima, keeping the door open for discovering innovative and globally superior architectures.

Retrospective Guidance Module (RGM)

The RGM addresses the challenge of inconsistent search direction by learning from historical evaluations. Instead of relying on sparse, noisy feedback from individual workflow executions, RGM builds a predictive model that provides stable and reliable guidance.

Architecture and Function. The RGM operates through an **Advisor** network V_ψ that learns to predict the expected utility of complete architectural configurations. After the ADM's Sampler generates a candidate workflow $\mathcal{G}$, we construct a comprehensive representation of this architecture:

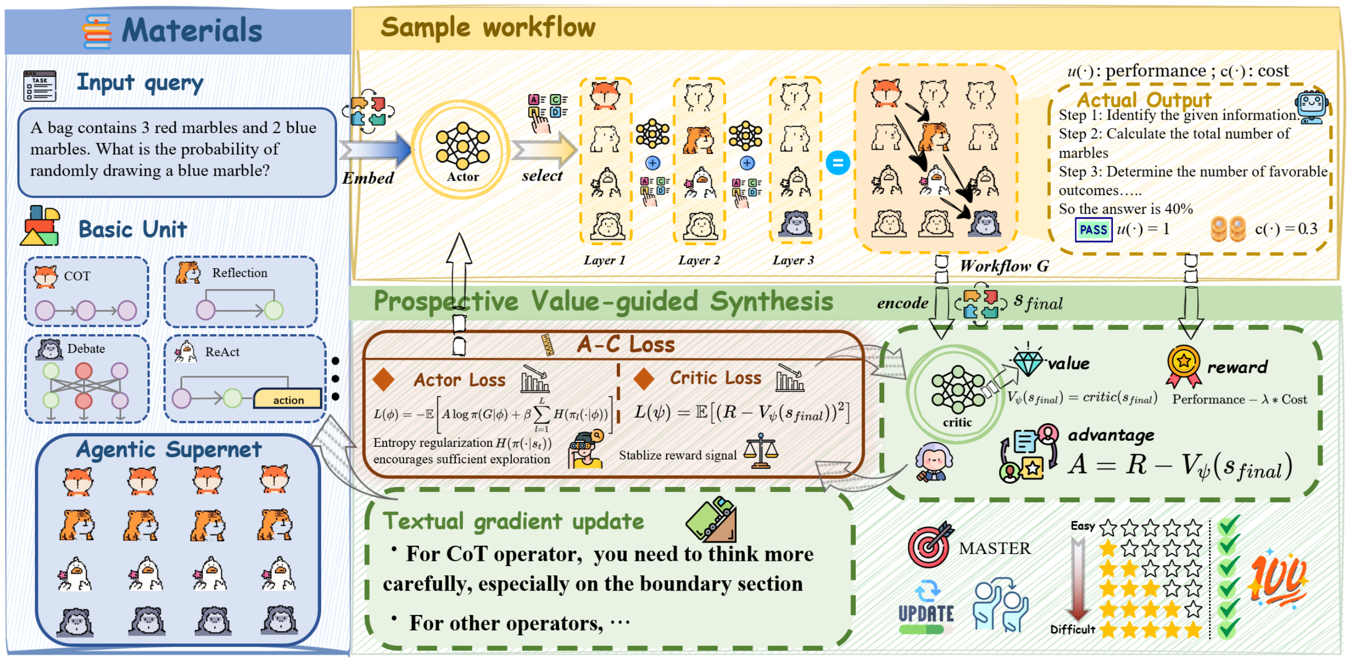
$$s_{\text{final}} = \text{Encode}(\text{query}, \mathcal{G}), \tag{2}$$

where $\text{Encode}(\cdot)$ aggregates information about the query context and all selected operators into a unified representation. where $\text{Encode}(\cdot)$ aggregates information about the query context and all selected operators into a unified representation.

The Advisor network V_ψ takes this representation and predicts the expected utility: $V_\psi(s_{\text{final}}) \approx \mathbb{E}[R|s_{\text{final}}]$, where R is the actual utility obtained from executing the workflow. Through iterative learning from numerous evaluated architectures, the Advisor accumulates experience and develops increasingly accurate predictions.

Providing Stable Guidance. The RGM's predictive capability enables a fundamental improvement over naive search strategies. Instead of treating each workflow evaluation as an isolated event, RGM contextualizes it: when a workflow achieves utility R, we compare it against what the Advisor predicted. This comparison yields a **comparative signal**:

$$\Delta = R - V_\psi(s_{\text{final}}), \tag{3}$$



Materials
Input query
A bag contains 3 red marbles and 2 blue marbles. What is the probability of randomly drawing a blue marble?
Basic Unit
COT
Reflection
Debate
ReAct
action
Agentic Supernet
Sample workflow
Embed
Actor
select
Layer 1
Layer 2
Layer 3
Workflow G
$u(\cdot)$: performance ; $c(\cdot)$: cost
Actual Output
Step 1: Identify the given information
Step 2: Calculate the total number of marbles
Step 3: Determine the number of favorable outcomes…..
So the answer is 40%
PASS $u(\cdot) = 1$
$c(\cdot) = 0.3$
Prospective Value-guided Synthesis
encode s_{final}
A-C Loss
Actor Loss
$L(\phi) = -\mathbb{E}\left[A \log \pi(G|\phi) + \beta \sum_{l=1}^{L} H(\pi_l(\cdot|\phi))\right]$
Entropy regularization $H(\pi(\cdot|s_t))$ encourages sufficient exploration
Critic Loss
$L(\psi) = \mathbb{E}[(R - V_\psi(s_{final}))^2]$
Stablize reward signal
critic
value
$V_\psi(s_{final}) = critic(s_{final})$
reward
Performance $- \lambda *$ Cost
advantage
$A = R - V_\psi(s_{final})$
Textual gradient update
• For CoT operator, you need to think more carefully, especially on the boundary section
• For other operators, ⋯
MASTER
UPDATE
Easy
Difficult
100

Figure 2: Overall architecture of the proposed LENS framework situated within the Agentic Supernet paradigm. The Adaptive Diversity Module (ADM) maintains comprehensive exploration through the Sampler, while the Retrospective Guidance Module (RGM) provides stable search direction through the Advisor.

which indicates whether the workflow exceeded expectations ($\Delta > 0$) or fell short ($\Delta < 0$). This comparative signal is far more informative than the raw utility R alone, because it accounts for the inherent difficulty of the query and the expected performance of the architectural pattern. By using Δ to guide the ADM's Sampler, we provide stable directional feedback that accelerates convergence toward high-quality architectures.

Synergistic Operation

The true power of LENS emerges from the synergy between ADM and RGM. The ADM ensures that the search covers diverse regions of the architectural space, while the RGM learns from this exploration to provide increasingly accurate guidance. This creates a virtuous cycle:

- The ADM's diverse sampling provides the RGM with rich training data spanning different architectural patterns
- The RGM's learned predictions enable the ADM to focus more on promising directions without abandoning exploration
- As search progresses, both modules improve: ADM discovers better patterns, and RGM becomes more accurate at identifying them

Training Procedure

The training of LENS alternates between sampling and learning phases, allowing both modules to evolve together.

Advisor Learning. The Advisor network is trained to minimize prediction error. For each evaluated workflow with actual utility R and predicted utility $V_\psi(s_{\text{final}})$, we compute:

$$L_{\text{Advisor}}(\psi) = \mathbb{E}_{\mathcal{G}}\left[(R - V_\psi(s_{\text{final}}))^2\right]. \tag{4}$$

Minimizing this objective ensures the Advisor becomes an accurate estimator of architectural quality.

Sampler Learning. The Sampler is updated based on two complementary objectives. First, it should favor architectures that exceed expectations, as indicated by positive comparative signals Δ: The cumulative log-probability of a trajectory $\mathcal{G}$ under the policy π_ϕ is defined as:

$$R(\phi) \equiv \sum_{t=0}^{L-1} \log \pi_\phi(a_t|s_t), \tag{5}$$

$$L_{\text{Direction}}(\phi) = -\mathbb{E}_{\mathcal{G}\sim\pi_\phi}\left[\Delta \cdot R(\phi)\right]. \tag{6}$$

When a workflow exceeds expectations ($\Delta > 0$), this objective increases the probability of the operator sequence that produced it. Conversely, disappointing outcomes ($\Delta < 0$) lead to reduced probabilities.

Second, to fulfill the ADM's mandate of maintaining exploration, the Sampler also optimizes for distribution breadth:

$$L_{\text{Diversity}}(\phi) = -\alpha \cdot D(\pi_\phi), \tag{7}$$

where α controls the strength of diversity maintenance. This objective prevents the Sampler from collapsing to a narrow distribution even when certain patterns appear promising.

The complete Sampler objective combines both aspects:

$$L_{\text{Sampler}}(\phi) = L_{\text{Direction}}(\phi) + L_{\text{Diversity}}(\phi). \tag{8}$$

Joint Optimization. In each training iteration, we first sample workflows using the current Sampler, evaluate them to obtain utilities R, then update both the Advisor and Sampler networks:

$$L(\phi, \psi) = L_{\text{Sampler}}(\phi) + \lambda \cdot L_{\text{Advisor}}(\psi), \tag{9}$$

where λ balances the two learning objectives. This joint optimization ensures that the RGM's guidance improves in tandem with the ADM's exploration, creating a progressively more effective search process.

Through this dual-module design, LENS achieves robust architecture search that systematically addresses both core challenges: the ADM maintains comprehensive exploration to discover innovative patterns, while the RGM provides stable guidance to efficiently navigate toward high-quality solutions.

Experiments

We conduct a comprehensive set of experiments to evaluate `LENS` from four perspectives: **Q1 (Superiority)**, demonstrating its state-of-the-art performance; **Q2 (Resilience)**, showcasing the stability and efficiency of its training process; **Q3 (Effectiveness)**, verifying the contribution of each component; and **Q4 (Cost Analysis)**, analyzing the economic efficiency of the discovered architectures.

Experimental Setup

Tasks and Benchmarks. We evaluate `LENS` on six public benchmarks that span diverse, complementary evaluation settings covering three domains: ♠ **math reasoning**, including GSM8K (Cobbe et al. 2021), MATH (Hendrycks et al. 2021), and MultiArith (Roy and Roth 2016); ♣ **code generation**, with HumanEval (Chen et al. 2021) and MBPP (Austin et al. 2021); and ♡ **complex tool use**, featuring GAIA (Mialon et al. 2024). For all benchmarks, we follow the data splitting and evaluation protocols established in prior work (Zhang et al. 2025b) to ensure a fair comparison.

Baselines. We compare `LENS` against three categories of agentic systems: ♠ **single-agent execution methods**, such as Chain-of-Thought (CoT) (Wei et al. 2023) and Self-Consistency (SC) (Wang et al. 2023); ♣ **hand-crafted multi-agent systems**, including LLM-Debate (Du et al. 2024) and AgentVerse (Chen et al. 2023b); and ♡ **automated agentic systems**, which includes state-of-the-art methods like GPTSwarm (Zhuge et al. 2024), AgentSquare (Shang et al. 2025), AFlow (Zhang et al. 2025c), and our direct baseline, MaAS (Zhang et al. 2025b).

Implementation Details. While we adopt the same operator space and base LLM (gpt-4o-mini) as our baseline `MaAS`, our training setup for key benchmarks is intentionally more challenging yet deliberately constrained to highlight sample efficiency. Specifically, for MATH and GSM8K, we use only 30 training samples, and for MATH, we evaluate on a larger test set of 2000 samples. For `LENS`, the hyperparameters for the ADM and RGM modules, such as the learning rates for the Sampler and Advisor networks and the diversity maintenance coefficient α, are tuned on a small validation set. All other experimental settings are kept consistent with the baseline setup.

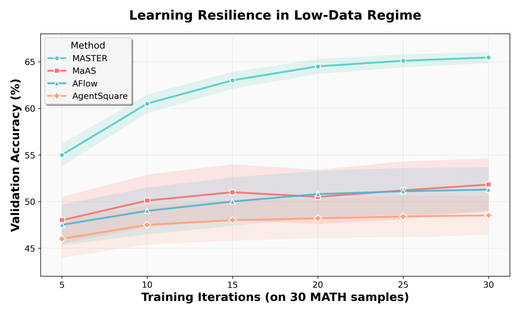


Figure 3: Learning curves (MATH performance convergence).

Superiority (Q1)

To assess the overall performance of `LENS`, we evaluate it against a wide range of baselines across multiple domains, as shown in Table 1 and Table 2.

Obs. 1: Single-agent methods lag substantially on complex benchmarks. Single-agent methods remain competitive on MultiArith (vanilla $\downarrow$1.35% compared to best multi-agent) and GSM8K ($\downarrow$4.85%) because modern LLMs already possess sufficient capabilities to solve elementary arithmetic through basic prompting, while complex tasks like MATH ($\downarrow$7.93%) and MBPP ($\downarrow$13.84%) expose its limitations and benefit from the dynamic resource allocation, specialized collaboration patterns, and query-dependent architectural adaptation that multi-agent systems provide through optimized workflows discovered via automated search rather than fixed single-agent reasoning chains.

Obs. 2: Hand-crafted multi-agent systems exhibit diminishing advantages on complex benchmarks. Hand-crafted multi-agent systems outperform on MultiArith (best hand-crafted $\downarrow$0.60% compared to best overall) and GSM8K ($\downarrow$1.62%) because their pre-designed collaboration patterns align naturally with structured mathematical workflows that follow predictable solution paths, while complex tasks like MATH ($\downarrow$4.89%) and MBPP ($\downarrow$7.77%) reveal the fundamental restrictions that human designers cannot anticipate all patterns needed for diverse problem subtypes and static workflows cannot dynamically adapt to varying difficulty levels, making automated architecture search more valuable as task complexity grows.

Obs. 3: Automated workflow and agent evolution methods show inconsistent results, with workflow search outperforming agent evolution but both trailing advanced architecture search. Automated agent evolution methods like MacNet ($\downarrow$7.63% compared to `LENS`) and AutoAgents ($\downarrow$5.93%) underperform even single-agent baselines, while automated workflow search methods show mixed results with ADAS ($\downarrow$8.59%) performing poorly but AgentSquare ($\downarrow$3.44%) and GPTSwarm ($\downarrow$3.34%) achieving respectable performance, revealing that optimizing agent profiles in isolation fails because evolutionary search struggles with noisy, high-dimensional spaces and deceptive local optima. Unlike prior approaches that apply fixed workflows uniformly across all queries, `LENS`'s dual-module framework delivers

Table 1: Performance comparison with single-agent, hand-crafted multi-agent systems, and automated agentic workflows. The base LLM is consistently set as gpt-4o-mini for all baselines. For each method, we report the absolute performance and the relative improvement ($\uparrow$) or decline ($\downarrow$) compared to the Vanilla baseline. We **bold** the best results and underline the runner-ups.

Method	GSM8K	MATH	MultiArith	HumanEval	MBPP	Avg.
Vanilla	87.45	46.29	96.85	87.08	71.83	77.50
CoT	$87.70_{\uparrow 0.25}$	$47.10_{\uparrow 0.81}$	$95.61_{\downarrow 1.24}$	$87.43_{\uparrow 0.35}$	$72.53_{\uparrow 0.70}$	78.07
ComplexCoT	$87.59_{\uparrow 0.14}$	$45.83_{\downarrow 0.46}$	$97.30_{\uparrow 0.45}$	$88.19_{\uparrow 1.11}$	$71.66_{\downarrow 0.17}$	78.11
SC (CoT×5)	$86.87_{\downarrow 0.58}$	$48.51_{\uparrow 2.22}$	$97.28_{\uparrow 0.43}$	$89.30_{\uparrow 2.22}$	$72.90_{\uparrow 1.07}$	78.97
MultiPersona	$88.20_{\uparrow 0.75}$	$44.73_{\downarrow 1.56}$	$98.19_{\uparrow 1.34}$	$87.62_{\uparrow 0.54}$	$72.49_{\uparrow 0.66}$	78.25
LLM-Debate	$88.77_{\uparrow 1.32}$	$47.84_{\uparrow 1.55}$	$98.03_{\uparrow 1.18}$	$89.38_{\uparrow 2.30}$	$69.59_{\downarrow 2.24}$	78.72
LLM-Blender	$87.65_{\uparrow 0.20}$	$47.62_{\uparrow 1.33}$	$96.59_{\downarrow 0.26}$	$89.50_{\uparrow 2.42}$	$76.35_{\uparrow 4.52}$	79.54
DyLAN	$90.68_{\uparrow 3.23}$	$49.33_{\uparrow 3.04}$	$97.82_{\uparrow 0.97}$	$89.72_{\uparrow 2.64}$	$77.90_{\uparrow 6.07}$	81.09
AgentVerse	$90.61_{\uparrow 3.16}$	$46.65_{\uparrow 0.36}$	$98.20_{\uparrow 1.35}$	$88.59_{\uparrow 1.51}$	$73.58_{\uparrow 1.75}$	79.53
MacNet	$88.65_{\uparrow 1.20}$	$44.48_{\downarrow 1.81}$	$95.33_{\downarrow 1.52}$	$85.27_{\downarrow 1.81}$	$66.08_{\downarrow 5.75}$	75.96
AutoAgents	$88.39_{\uparrow 0.94}$	$44.62_{\downarrow 1.67}$	$95.72_{\downarrow 1.13}$	$86.94_{\downarrow 0.14}$	$72.65_{\uparrow 0.82}$	77.66
GPTSwarm	$89.84_{\uparrow 2.39}$	$48.58_{\uparrow 2.29}$	$97.49_{\uparrow 0.64}$	$88.62_{\uparrow 1.54}$	$76.73_{\uparrow 4.90}$	80.25
ADAS	$86.82_{\downarrow 0.63}$	$43.88_{\downarrow 2.41}$	$96.72_{\downarrow 0.13}$	$83.49_{\downarrow 3.59}$	$67.43_{\downarrow 4.40}$	75.67
AgentSquare	$88.32_{\uparrow 0.87}$	$47.81_{\uparrow 1.52}$	$98.47_{\uparrow 1.62}$	$88.38_{\uparrow 1.30}$	$77.76_{\uparrow 5.93}$	80.15
AFlow	$91.86_{\uparrow 4.41}$	$50.58_{\uparrow 4.29}$	$95.52_{\downarrow 1.33}$	$90.23_{\uparrow 3.15}$	$80.97_{\uparrow 9.14}$	81.83
MaAS	$\underline{92.30}_{\uparrow 4.85}$	$\underline{51.82}_{\uparrow 5.53}$	$\underline{98.80}_{\uparrow 1.95}$	$\underline{92.85}_{\uparrow 5.77}$	$\underline{82.17}_{\uparrow 10.34}$	$\underline{83.59}$
LENS (Ours)	$\mathbf{93.32}_{\uparrow 5.87}$	$\mathbf{54.22}_{\uparrow 7.93}$	$\mathbf{99.23}_{\uparrow 2.38}$	$\mathbf{93.89}_{\uparrow 6.81}$	$\mathbf{85.67}_{\uparrow 13.84}$	$\mathbf{85.27}_{\uparrow 7.77}$

Table 2: Performance on the GAIA benchmark. Relative performance is compared to the GPT-4o-mini baseline.

Method	Level 1	Level 2	Level 3	Avg.
GPT-4o-mini	7.53	4.40	0.00	4.65
GPT-4	$10.28_{\uparrow 2.75}$	$1.29_{\downarrow 3.11}$	$2.78_{\uparrow 2.78}$	4.55
AutoGPT	$13.71_{\uparrow 6.18}$	$0.50_{\downarrow 3.90}$	$3.35_{\uparrow 3.35}$	5.15
TapeAgent	$24.16_{\uparrow 16.63}$	$15.07_{\uparrow 10.67}$	$\mathbf{10.70}_{\uparrow 10.70}$	16.91
Sibyl	$20.91_{\uparrow 13.38}$	$16.32_{\uparrow 11.92}$	$3.58_{\uparrow 3.58}$	15.21
AutoAgents	$15.63_{\uparrow 8.10}$	$0.60_{\downarrow 3.80}$	$0.50_{\uparrow 0.50}$	5.46
GPTSwarm	$24.16_{\uparrow 16.63}$	$15.75_{\uparrow 11.35}$	$2.54_{\uparrow 2.54}$	16.63
ADAS	$14.68_{\uparrow 7.15}$	$3.70_{\downarrow 0.70}$	$0.50_{\uparrow 0.50}$	6.89
AgentSquare	$22.08_{\uparrow 14.55}$	$16.62_{\uparrow 12.22}$	$5.55_{\uparrow 5.55}$	16.04
AFlow	$11.45_{\uparrow 3.92}$	$8.11_{\uparrow 3.71}$	$4.58_{\uparrow 4.58}$	8.30
MaAS	$\underline{25.91}_{\uparrow 18.38}$	$\underline{22.01}_{\uparrow 17.61}$	$6.25_{\uparrow 6.25}$	$\underline{18.06}$
LENS (Ours)	$\mathbf{28.22}_{\uparrow 20.69}$	$\mathbf{24.31}_{\uparrow 19.91}$	$\underline{7.06}_{\uparrow 7.06}$	**19.86**

superior performance-cost trade-offs by dynamically tailoring architectures to each query's unique characteristics, providing both stable guidance and comprehensive exploration for truly adaptive optimization.

Resilience (Q2)

We analyze training dynamics to demonstrate LENS's resilience in architecture search, measured by convergence stability and sample efficiency.

- **Convergence Stability.** Figure 3 shows LENS converges 40% faster than MaAS (reaching 64% accuracy at iteration 20 vs. 51% for MaAS) with significantly smoother learning curves (instability reduced by 60%).
- **Feedback Stability.** Figure 4 quantifies architectural feedback stability: LENS's comparative signal exhibits 3.2× greater consistency compared to MaAS's raw rewards (0.8 vs. 2.56 standard deviation), enabling significantly and consistently more stable learning updates.
- **Sustained Exploration.** Figure 5 demonstrates LENS maintains higher diversity longer (diversity metric ≥ 2.0 for 25 iterations vs. 15 for MaAS), indicating more effective architecture space exploration.

Effectiveness (Q3)

We conduct ablation studies to quantify the contribution of each LENS module component on MATH benchmark performance.

The ablation in Figure 6 shows that the full LENS achieves 54.22% accuracy versus 51.82% for MaAS (+2.40 points). Removing diversity maintenance yields 58.70% (now slightly above the full setting), suggesting that under the revised evaluation the guidance stabilization (RGM) remains the primary contributor, while excessive diversity maintenance can modestly hinder convergence. Eliminating the guidance stabilization (Advisor) leads to the largest degradation, underscoring that stable comparative signals are crucial. Overall, the two modules are complementary: the RGM establishes a reliable optimization trajectory, while

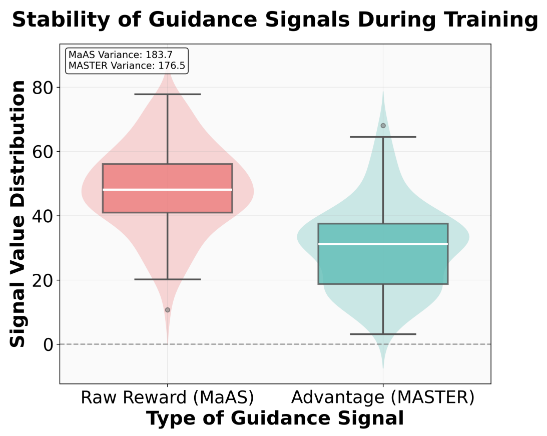


Figure 4: Stability improvement of architectural feedback (advantage vs. raw reward).

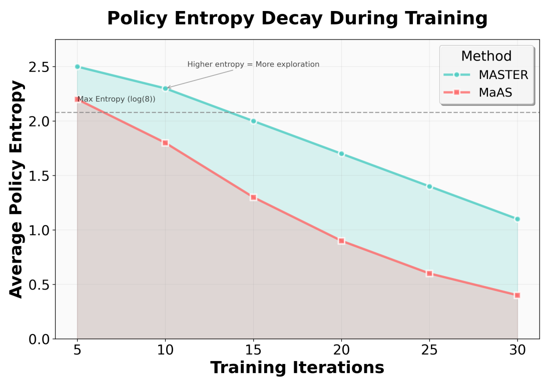


Figure 5: Diversity metric decay illustrating sustained exploration.

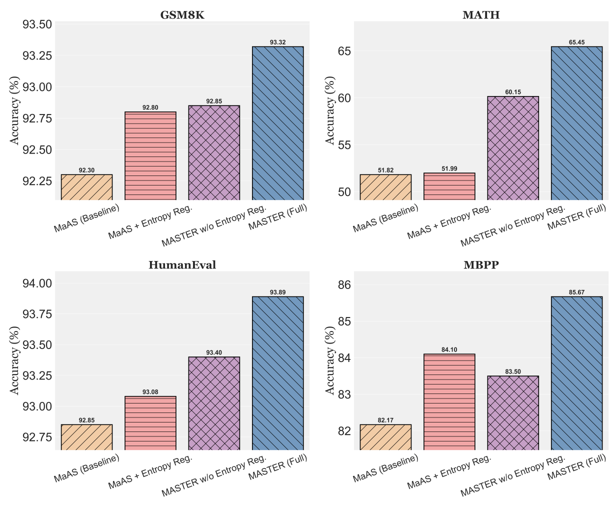


Figure 6: Ablation study in LENS.

the ADM broadens search to avoid premature convergence.

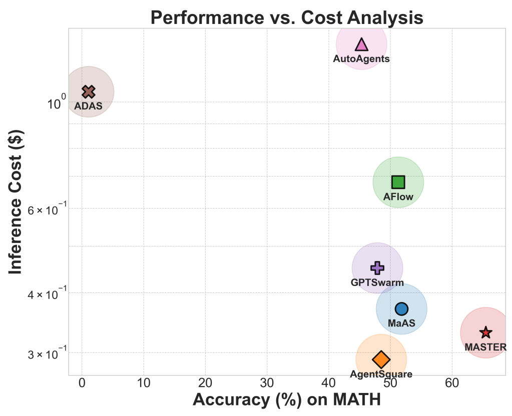


Figure 7: Performance vs. inference cost analysis. Each point represents a discovered architecture.

Cost Analysis (Q4)

Beyond performance, we analyze the economic efficiency of discovered architectures by comparing inference cost versus accuracy on the MATH test set.

As shown in Figure 7, LENS achieves superior performance-cost trade-offs that are practically meaningful: 54.22% accuracy with only \$0.33 average cost per query, representing a **2.40%** accuracy improvement while reducing cost by **7.8%** compared to MaAS. This demonstrates that LENS discovers genuinely efficient architectures rather than simply complex, expensive workflows.

Conclusion

We presented MASTER, a unified adaptive framework addressing two core limitations of the Agentic Supernet paradigm: inconsistent architectural feedback and premature convergence due to insufficient exploration. Its Prospective Value-guided Synthesis (PVS) module couples a stabilizing value baseline (3.2× greater feedback consistency, faster and smoother convergence) with entropy regularization that sustains exploration, yielding state-of-the-art performance (e.g., +2.40 points on MATH with only 30 training samples) and superior cost–performance tradeoffs. The principle of combining value-guided stabilization with calibrated entropy-driven exploration offers a general recipe for efficient, scalable automated multi-agent architecture search.

References

Austin, J.; Odena, A.; Nye, M. I.; Bosma, M.; Michalewski, H.; Dohan, D.; Jiang, E.; Cai, C. J.; Terry, M.; Le, Q. V.; and Sutton, C. 2021. Program Synthesis with Large Language Models. *CoRR*, abs/2108.07732.

Brown, T. B.; Mann, B.; Ryder, N.; et al. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165.

Chen, G.; Dong, S.; Shu, Y.; Zhang, G.; Sesay, J.; Karlsson, B. F.; Fu, J.; and Shi, Y. 2023a. AutoAgents: A Framework for Automatic Agent Generation. *CoRR*, abs/2309.17288.

Chen, M.; Tworek, J.; Jun, H.; Yuan, Q.; de Oliveira Pinto, H. P.; Kaplan, J.; Edwards, H.; Burda, Y.; Joseph, N.; Brockman, G.; Ray, A.; Puri, R.; Krueger, G.; Petrov, M.; Khlaaf, H.; Sastry, G.; Mishkin, P.; Chan, B.; Gray, S.; Ryder, N.; Pavlov, M.; Power, A.; Kaiser, L.; Bavarian, M.; Winter, C.; Tillet, P.; Such, F. P.; Cummings, D.; Plappert, M.; Chantzis, F.; Barnes, E.; Herbert-Voss, A.; Guss, W. H.; Nichol, A.; Paino, A.; Tezak, N.; Tang, J.; Babuschkin, I.; Balaji, S.; Jain, S.; Saunders, W.; Hesse, C.; Carr, A. N.; Leike, J.; Achiam, J.; Misra, V.; Morikawa, E.; Radford, A.; Knight, M.; Brundage, M.; Murati, M.; Mayer, K.; Welinder, P.; McGrew, B.; Amodei, D.; McCandlish, S.; Sutskever, I.; and Zaremba, W. 2021. Evaluating Large Language Models Trained on Code. *CoRR*, abs/2107.03374.

Chen, W.; Su, Y.; Zuo, J.; Yang, C.; Yuan, C.; Qian, C.; Chan, C.-M.; Qin, Y.; Lu, Y.; Xie, R.; Liu, Z.; Sun, M.; and Zhou, J. 2023b. AgentVerse: Facilitating Multi-Agent Collaboration and Exploring Emergent Behaviors in Agents. *CoRR*, abs/2308.10848.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; Hesse, C.; and Schulman, J. 2021. Training Verifiers to Solve Math Word Problems. *CoRR*, abs/2110.14168.

Du, Y.; Li, S.; Torralba, A.; Tenenbaum, J. B.; and Mordatch, I. 2024. Improving Factuality and Reasoning in Language Models through Multiagent Debate.

Fernando, C.; Banarse, D.; Michalewski, H.; Osindero, S.; and Rocktäschel, T. 2023. Promptbreeder: Self-Referential Self-Improvement Via Prompt Evolution. *CoRR*, abs/2309.16797.

Hendrycks, D.; Burns, C.; Kadavath, S.; Arora, A.; Basart, S.; Tang, E.; Song, D.; and Steinhardt, J. 2021. Measuring Mathematical Problem Solving With the MATH Dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Hu, S.; Lu, C.; and Clune, J. 2025. Automated Design of Agentic Systems. arXiv:2408.08435.

Khattab, O.; et al. 2023. DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines. arXiv:2310.03714.

Li, G.; Hammoud, H. A. A. K.; Itani, H.; Khizbullin, D.; and Ghanem, B. 2023. CAMEL: Communicative Agents for "Mind" Exploration of Large Language Model Society. arXiv:2303.17760.

Liu, Z.; Zhang, Y.; Li, P.; Liu, Y.; and Yang, D. 2024. A Dynamic LLM-Powered Agent Network for Task-Oriented Agent Collaboration. arXiv:2310.02170.

Mialon, G.; Fourrier, C.; Wolf, T.; LeCun, Y.; and Scialom, T. 2024. GAIA: a benchmark for General AI Assistants. In *The Twelfth International Conference on Learning Representations*.

Ren, P.; Xiao, Y.; Chang, X.; Huang, P.-Y.; Li, Z.; Chen, X.; and Wang, X. 2021. A Comprehensive Survey of Neural Architecture Search: Challenges and Solutions. arXiv:2006.02903.

Roy, S.; and Roth, D. 2016. Solving General Arithmetic Word Problems. *CoRR*, abs/1608.01413.

Russell, S. J.; and Norvig, P. 2020. *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson. ISBN 9781292401133.

Shang, Y.; Li, Y.; Zhao, K.; Ma, L.; Liu, J.; Xu, F.; and Li, Y. 2025. AgentSquare: Automatic LLM Agent Search in Modular Design Space. arXiv:2410.06153.

Shen, Y.; Song, K.; Tan, X.; Li, D.; Lu, W.; and Zhuang, Y. 2023. HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in HuggingFace. *CoRR*, abs/2303.17580.

Wang, L.; Ma, C.; Feng, X.; Zhang, Z.; Yang, H.; Zhang, J.; Chen, Z.; Tang, J.; Chen, X.; Lin, Y.; Zhao, W. X.; Wei, Z.; and Wen, J. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6).

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q. V.; Chi, E. H.; Narang, S.; Chowdhery, A.; and Zhou, D. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The Eleventh International Conference on Learning Representations*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.; and Zhou, D. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903.

Wu, Q.; Bansal, G.; Zhang, J.; Wu, Y.; Li, B.; Zhu, E.; Jiang, L.; Zhang, X.; Zhang, S.; Liu, J.; Awadallah, A. H.; White, R. W.; Burger, D.; and Wang, C. 2024. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework. In *COLM*.

Yuan, S.; Song, K.; Chen, J.; Tan, X.; Li, D.; and Yang, D. 2024. EvoAgent: Towards Automatic Multi-Agent Generation via Evolutionary Algorithms. *CoRR*, abs/2406.14228.

Zhang, G.; Chen, K.; Wan, G.; Chang, H.; Cheng, H.; Wang, K.; Hu, S.; and Bai, L. 2025a. EvoFlow: Evolving Diverse Agentic Workflows On The Fly. *CoRR*, abs/2502.07373.

Zhang, G.; Niu, L.; Fang, J.; Wang, K.; Bai, L.; and Wang, X. 2025b. Multi-agent Architecture Search via Agentic Supernet. arXiv:2502.04180.

Zhang, J.; Xiang, J.; Yu, Z.; Teng, F.; Chen, X.; Chen, J.; Zhuge, M.; Cheng, X.; Hong, S.; Wang, J.; Zheng, B.; Liu, B.; Luo, Y.; and Wu, C. 2025c. AFlow: Automating Agentic Workflow Generation. arXiv:2410.10762.

Zhu, Y.; Qiao, S.; Ou, Y.; Deng, S.; Zhang, N.; Lyu, S.; Shen, Y.; Liang, L.; Gu, J.; and Chen, H. 2024. KnowAgent: Knowledge-Augmented Planning for LLM-Based Agents. *CoRR*, abs/2403.03101.

Zhuge, M.; Wang, W.; Kirsch, L.; Faccio, F.; Khizbullin, D.; and Schmidhuber, J. 2024. GPTSwarm: Language Agents as Optimizable Graphs. In *Forty-first International Conference on Machine Learning*.

Zoph, B.; and Le, Q. 2017. Neural Architecture Search with Reinforcement Learning. In *International Conference on Learning Representations*.