LLM-Guided Analogical Reasoning Particle Swarm Optimization for Neural Architecture Search

1st Tianshui Li
Graduate School of Information Science and Technology
The University of Tokyo
Tokyo, Japan
tianshui@g.ecc.u-tokyo.ac.jp

2nd Keiki Takadama
Graduate School of Information Science and Technology
The University of Tokyo
Tokyo, Japan
takadama@g.ecc.u-tokyo.ac.jp

3rd Hitoshi Iba
Graduate School of Information Science and Technology
The University of Tokyo
Tokyo, Japan
iba@iba.t.u-tokyo.ac.jp

***Abstract*—The integration of Large Language Models (LLMs) into Neural Architecture Search (NAS) has introduced a new paradigm that leverages LLMs' domain knowledge to generate valid candidate architectures efficiently. However, a key limitation of existing methods is that their mutation operations often lack explicit guidance from the evolutionary history. They are mainly driven by the LLM's internal pre-trained priors without explicitly analyzing the structural trends within the optimization landscape. This mismatch often leads to suggestions that are poorly aligned with the problem-specific search landscape and can reduce search efficiency. To address this issue, we propose LLM-Guided Analogical Reasoning Particle Swarm Optimization (LARPSO). The proposed framework integrates the learning dynamics of particle swarm optimization into LLM-driven mutation by providing the model with explicit improvement trajectories derived from the swarm search history. This mechanism encourages the LLM to infer structural improvement patterns and apply analogous transformations to the current architecture, which mitigates stagnation and promotes more directed exploration. We evaluate LARPSO on the NAS-Bench-201 benchmark across three datasets. Experimental results show that LARPSO consistently outperforms representative LLM-based NAS baselines and remains competitive with other state-of-the-art NAS methods. The source code of LARPSO is publicly available at https://github.com/LiTianshui/LARPSO.**

***Index Terms*—evolutionary neural architecture search, particle swarm optimization, large language model, evolutionary computation**

I. Introduction

Neural Architecture Search (NAS) is an active research area that aims to automate the design of neural network architectures for specific tasks and datasets [3]. Over the past decade, a wide range of NAS algorithms have been proposed, and many of them have demonstrated strong potential in discovering high-quality architectures. Furthermore, several studies have shown that NAS can discover architectures that surpass the performance of those manually designed by human experts [10] [11] [12].

With the rapid advancement of Large Language Models (LLMs) [6], a new paradigm of LLM-based NAS has emerged [7], represented by methods such as GENIUS [8] and LLMatic [9]. A primary advantage of these approaches is their ability to leverage LLMs' domain knowledge to generate valid candidate architectures efficiently. GENIUS utilizes the generative capabilities of LLM as a black-box optimizer to navigate the search space iteratively, whereas LLMatic combines code-generating models with Quality-Diversity method (QD) [29] to maintain diverse architectural archives. These prior approaches often provide the model with limited structured information regarding architectural evolution. Consequently, the generated suggestions may not align well with the local structure of a particular search space, frequently causing premature stagnation or inefficient exploration.

To address this issue, we propose LLM-Guided Analogical Reasoning Particle Swarm Optimization (LARPSO). The proposed method integrates the learning dynamics of particle swarm optimization (PSO) into LLM-driven mutation by providing the model with explicit improvement trajectories derived from the swarm search history. This mechanism encourages the LLM to infer structural improvement patterns and apply analogous transformations to the current architecture, which mitigates stagnation and promotes more directed exploration.

Our contributions are summarized as follows:

1) We identify a critical limitation of existing LLM-based NAS methods in which architecture mutations are often insufficiently informed by evolutionary history and can rely heavily on pre-trained priors, leading to less targeted optimization.
2) We introduce the LARPSO method which combines the domain knowledge and reasoning capability of LLM with PSO-inspired search dynamics. This design encourages the LLM to infer structural improvement patterns and apply analogous transformations to the current so-

lution, which mitigates stagnation and promotes more directed exploration.

3) We evaluate LARPSO on the NAS-Bench-201 benchmark across three datasets. The results demonstrate that LARPSO consistently outperforms representative LLM-based NAS baselines and remains competitive with other state-of-the-art NAS methods.

The remainder of the article is organized as follows. Section II reviews the related work such as PSO, NAS, and NAS benchmarks. Section III details the proposed LARPSO algorithm, including the overall structure and prompt design. Section IV presents the experimental settings and comparison results. Section V provides a comprehensive analysis of the proposed algorithm. Finally, Section VI concludes the paper and outlines possible future research directions.

II. Related Work

A. Particle Swarm Optimizer

Particle Swarm Optimization (PSO) was originally proposed by Kennedy and Eberhart in 1995 [1], inspired by the collective behavior of bird flocks and fish schools. In this algorithm, a group of particles searches for optimal solutions by moving through the problem space. Each particle represents a candidate solution and adjusts its position iteratively based on both personal experience and information shared by the swarm. This update mechanism enables particles to explore the search space by balancing personal and social experiences. The simplicity and effectiveness of this approach have made PSO a widely used optimization method in both academic and industrial applications.

Recently, LMPSO [27] incorporated LLMs into the PSO framework by reformulating the velocity term as natural language guidance within a structured prompt. This approach demonstrated competitive performance on the traveling salesman problem (TSP) [28] by generating new candidate solutions conditioned on the current solution as well as personal and global best solutions. Nevertheless, it primarily relies on direct solution generation without explicitly inferring underlying improvement patterns from the search history. In contrast, our proposed LARPSO emphasizes trend extraction from paired historical references and applies an analogical transformation to the mutation operation.

B. Neural Architecture Search

Neural Architecture Search (NAS) studies algorithmic ways to discover neural network architectures that perform well within a predefined search space under practical constraints. Prior work covers several major families of methods. Black-box approaches, such as reinforcement learning [2] and evolutionary search [5], are flexible but can require many evaluations. Gradient-based approaches improve efficiency by optimizing a continuous relaxation of discrete architectural choices, yet their results can be sensitive to the relaxation and training dynamics [4]. Weight-sharing and one-shot approaches reduce cost by reusing parameters across candidates, but the induced proxy ranking may differ from the ranking under full training [24] [25].

Recent progress in LLMs has introduced a new paradigm that uses LLMs to propose architectures. In this framework, architectures are represented as structured strings or executable code. The LLM produces candidates conditioned on task descriptions and observed outcomes. This formulation reduces the need for handcrafted search operators and facilitates the incorporation of prior design knowledge through prompting.

GENIUS [8] leverages the generative capabilities of the LLM to iteratively generate candidate networks and refine them using performance feedback. This makes the method conceptually simple and less dependent on specialized differentiable formulations. LLMatic [9] integrates code-generating language models with QD to discover architectures that are both diverse and high-performing. A primary advantage of this approach lies in its dual-archive cooperative mechanism which utilizes the MAP-Elites algorithm [26] to simultaneously maintain archives for networks and prompts. This ensures the diversity and robustness of the solutions.

Despite these strengths, a critical limitation shared by these algorithms is that their mutation operations often lack explicit guidance from the evolutionary history. Although GENIUS incorporates performance feedback, these methods are mainly driven by the internal pre-trained priors, which may lead to inefficient exploration or stagnation in a specific search space. This limitation motivates approaches that convert search history into explicit guidance for LLM-driven mutation. Such guidance can improve targeting and reliability during architecture evolution.

C. NAS Benchmarks

NAS benchmarks are standardized evaluation tools designed to make NAS research faster and more reproducible [20]. In this work, we use NAS-Bench-201 [21], a widely adopted tabular benchmark. It defines a cell-based search space with four nodes and five candidate operations per edge. This formulation results in a total of 15,625 unique architectures. Each architecture is trained with a unified protocol and evaluated on multiple datasets, including CIFAR-10, CIFAR-100 [22], and ImageNet-16-120 [23]. This enables efficient and reproducible benchmarking through table lookups rather than expensive training.

III. Proposed Method

While LLMs encode architectural patterns from their domain expertise, such generalized representations may not perfectly align with the specific constraints of individual search spaces. Effectively utilizing LLMs for NAS requires bridging the gap between their general knowledge and the specific problem context.

To address this challenge, we propose LARPSO, a framework that integrates the evolutionary dynamics of PSO with the reasoning capabilities of LLMs. Unlike traditional methods that are mainly driven by internal priors, LARPSO leverages the swarm's search history to provide context-aware feedback,

allowing the LLM to infer optimization directions dynamically.

In standard PSO, a particle updates its position based on velocity vectors derived from the difference between its current position and the personal or global best positions. However, since neural architectures reside in a discrete search space where vector subtraction is undefined. LARPSO replaces the concept of velocity with LLM-driven analogical mutation. Specifically, the LLM is prompted to analyze the structural transition between a weaker historical reference solution and a better one. By extracting the underlying improvement pattern from this comparison, it applies a logically analogous transformation to the current particle.

Algorithm 1 presents the overall procedure. In addition to the standard personal best and global best records, LARPSO maintains personal second-best and global second-best solutions. These auxiliary records ensure valid comparative pairs are always available to help the LLM infer structural improvements. During the cognitive learning step, the LLM extracts a local optimization trend by comparing the current particle with personal best record. Leveraging this observed trend, the LLM performs a directed mutation on the current solution to generate an intermediate solution.

Similarly, in the social learning step, the global exemplars serve as the comparative basis. The LLM analyzes the global search history to extract a broader improvement pattern. It then applies this insight to mutate the intermediate solution, resulting in the final new solution. The specific prompt details utilized to facilitate this reasoning are illustrated in Fig. 1.

Algorithm 1 Framework of the proposed LARPSO

Require: Population Size N, Max Generations T
Ensure: Global Best Solution g_{best}
Notation: p_{best}: personal best; p_{2nd}: personal 2^{nd} best; g_{best}: global best; g_{2nd}: global 2^{nd} best
Phase 1: Initialization
Initialize swarm $P = \{x_1, \ldots, x_N\}$ randomly and evaluate.
Set $p_{best,i} \leftarrow x_i$ and $p_{2nd,i} \leftarrow \emptyset$ for all $i = 1 \ldots N$.
Calculate g_{best} and g_{2nd} from the initial population P.
Phase 2: Optimization Loop
$t \leftarrow 0$
while $t < T$ **do**
  for all particle $x_i \in P$ **do**
    Step A: Cognitive Learning (Toward $\mathbf{p_{best}}$***)***
    if $x_i \neq p_{best,i}$ **then**
      $ref_{good} \leftarrow x_i$; $ref_{better} \leftarrow p_{best,i}$
      $x_{temp} \leftarrow \text{AM}(x_i, ref_{good}, ref_{better})$
    else if $p_{2nd,i} \neq \emptyset$ **then**
      $ref_{good} \leftarrow p_{2nd,i}$; $ref_{better} \leftarrow p_{best,i}$
      $x_{temp} \leftarrow \text{AM}(x_i, ref_{good}, ref_{better})$
    else
      $x_{temp} \leftarrow x_i$
    end if
    Step B: Social Learning (Toward $\mathbf{g_{best}}$***)***
    if $x_i \neq g_{best,i}$ **then**
      $ref_{good} \leftarrow x_i$; $ref_{better} \leftarrow g_{best}$
      $x_{new} \leftarrow \text{AM}(x_{temp}, ref_{good}, ref_{better})$
    else if $g_{2nd} \neq \emptyset$ **then**
      $ref_{good} \leftarrow g_{2nd}$; $ref_{better} \leftarrow g_{best}$
      $x_{new} \leftarrow \text{AM}(x_{temp}, ref_{good}, ref_{better})$
    else
      $x_{new} \leftarrow x_{temp}$
    end if
    Step C: Evaluation & Update
    Evaluate the fitness of new particle x_{new}.
    Update particle position $x_i \leftarrow x_{new}$.
    Update $p_{best,i}$, $p_{2nd,i}$, g_{best} and g_{2nd}.
  end for
  $t \leftarrow t + 1$
end while
return g_{best}

$\text{AM}(x, ref_{good}, ref_{better})$ denotes LLM-guided analogical mutation applied to target architecture x using ref_{good} and ref_{better} as the good and better references.

IV. Experimental Results

We evaluate LARPSO on NAS-Bench-201 benchmark [21]. This tabular benchmark enumerates 15,625 cell-based architectures and provides fully trained performance records for each candidate. This enables architecture evaluation by direct table lookup rather than repeated network training, which greatly reduces computational cost and improves reproducibility. Experiments are conducted on three standard datasets included in NAS-Bench-201, namely CIFAR-10, CIFAR-100 [22], and ImageNet-16-120 [23]. For all experiments across these datasets, we configure the population size to 5 and the maximum number of generations to 10. This setting results in a total budget of 50 architecture evaluations per run. We conduct three independent trials on each dataset to ensure statistical reliability. For LARPSO, we use Gemini 3 Pro as the LLM backend to perform the analogical mutation operator described in Section III. To encourage diverse structural proposals and prevent deterministic sequence generation, the sampling temperature is uniformly distributed between 0.2 and 1.8 for each mutation.

To prevent test set leakage during architecture selection, validation accuracy is used as the fitness signal throughout the search and as the criterion for selecting the final architecture. Test accuracy is used only for the final reporting of the selected architecture. We compare against two representative LLM-based NAS methods, GENIUS [8] and LLMatic [9]. Additionally, we include widely used NAS-Bench-201 baselines that achieve near-optimal results on this benchmark, including PC-DARTS [13], SNAS [14], iDARTS [15], GDAS [16], DRNAS [17], β-DARTS [18], and Λ-DARTS [19].

Table I presents the validation and test accuracies of LARPSO and the comparison methods on NAS-Bench-201. For LARPSO, we report the mean and standard deviation over three independent trials to ensure statistical robustness. The results demonstrate that our proposed LARPSO achieves superior performance across all three datasets when compared with two LLM-based NAS baselines. Furthermore, LARPSO exhibits strong performance relative to other state-of-the-art methods. Notably, it achieves the optimal results on the CIFAR-10 dataset.

In addition to final performance, we analyze the search efficiency and stability of the LARPSO. The convergence curves of LARPSO are illustrated in Fig. 2. These curves depict the progression of validation and test accuracy relative to the number of architecture evaluations. The solid lines represent the mean accuracy, while the shaded regions indicate

```
You are a Neural Architecture Search (NAS) Expert specializing in the NAS-Bench-201 search space.
Your Goal: Mutate the Current Architecture to create a superior candidate by combining:
1. Observed Trend: The structural improvement logic seen between Good Reference and Better Reference.
2. Domain Expertise: Your internal knowledge of what works best in NAS-Bench-201.
Search Space Definitions:
1. The architecture consists of 6 edges (operations).
2. Allowed Operations: {{none, skip_connect, nor_conv_1x1, nor_conv_3x3, avg_pool_3x3}}.
3. Format: |op~0|+|op~0|op~1|+|op~0|op~1|op~2|
Input Architectures:
1. Current Architecture: {current_arch_str}
2. Good Reference: {ref_good_str}
3. Better Reference: {ref_better_str}
Step-by-Step Reasoning Instructions:
1. Analyze the Trend: Compare Good vs Better. What changed?
2. Evaluate Current Architecture: Look at the Current Architecture. Does it suffer from too many
   parameters? Lack of connections?
3. Apply Hybrid Mutation: Apply the principle from step 1 to the Current Architecture, BUT modify it
   using your NAS expertise to suit the specific context of the Current Architecture.
Constraints (Strict Compliance Required):
1. Modification Limit: Change 1 to 2 operations in the Current Architecture. Keep the rest stable.
2. NO DUPLICATES: The output MUST be strictly different from Input Architectures.
3. Output Format: Return ONLY the architecture string. Do NOT provide explanations.
Return ONLY the new mutated architecture string in the standard format.
```

Fig. 1: Prompt design for LLM-guided analogical mutation.

TABLE I: Experimental results on NAS-Bench-201 benchmark across CIFAR-10, CIFAR-100, and ImageNet-16-120 datasets. The results for LARPSO are reported as the mean ± standard deviation over three independent trials. The optimal results represent the highest possible accuracy achievable within the search space for each dataset.

Method	CIFAR-10		CIFAR-100		ImageNet-16-120	
	Validation	Test	Validation	Test	Validation	Test
PC-DARTS [13]	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 [14]	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 [15]	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 [16]	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 [17]	91.55 ± 0.00	94.36 ± 0.00	$\mathbf{73.49 \pm 0.00}$	$\mathbf{73.51 \pm 0.00}$	$\mathbf{46.37 \pm 0.00}$	46.34 ± 0.00
β-DARTS [18]	91.55 ± 0.00	94.36 ± 0.00	$\mathbf{73.49 \pm 0.00}$	$\mathbf{73.51 \pm 0.00}$	$\mathbf{46.37 \pm 0.00}$	46.34 ± 0.00
Λ-DARTS [19]	91.55 ± 0.00	94.36 ± 0.00	$\mathbf{73.49 \pm 0.00}$	$\mathbf{73.51 \pm 0.00}$	$\mathbf{46.37 \pm 0.00}$	46.34 ± 0.00
GENIUS [8]	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 [9]	-	94.26 ± 0.13	-	71.62 ± 1.73	-	45.87 ± 0.96
LARPSO (Ours)	$\mathbf{91.61 \pm 0.00}$	$\mathbf{94.37 \pm 0.00}$	73.32 ± 0.31	73.10 ± 0.71	46.33 ± 0.09	$\mathbf{46.41 \pm 0.23}$
Optimal	91.61	94.37	73.49	73.51	46.73	47.31

the standard deviation across the three independent trials. As shown in Fig. 2, LARPSO exhibits rapid convergence on all three datasets, and the narrow width of the shaded regions confirms the stability of the LARPSO.

To validate that LARPSO performs directed mutation by extracting improvement patterns rather than random perturbations, we conduct a qualitative case study. Table II illustrates how a particle evolves through cognitive and social learning steps during a single generation in one CIFAR-10 trial. In the cognitive learning step, the reference transition suggests that replacing less informative operators with more informative ones can be beneficial, as the better reference replaces `nor_conv_1x1` and `none` with `nor_conv_3x3` and `skip_connect`. Guided by this trend, the mutation replaces `none` with `skip_connect` and upgrades `nor_conv_1x1` to `nor_conv_3x3` on the corresponding edges and keeps the other edges unchanged. This operation improves the validation accuracy from 89.53% to 91.08%. In the subsequent social learning step, the global reference indicates a further gain by increasing the representational capacity on the last edge, and the LLM proposes a consistent edit that replaces the terminal `skip_connect` with `nor_conv_1x1`, which results in an additional improvement from 91.08% to 91.61%.

V. Discussion

The experimental results in Table I and Fig. 2 support the effectiveness of LARPSO. The results indicate that integrating particle swarm dynamics with LLM-based reasoning significantly improves search quality relative to existing LLM-based NAS baselines.

On CIFAR-10, LARPSO achieves a test accuracy of $94.37\% \pm 0.00$, which matches the optimal performance defined by the NAS-Bench-201 benchmark. In contrast, GENIUS and LLMatic achieved 93.79% and 94.26% respectively. The

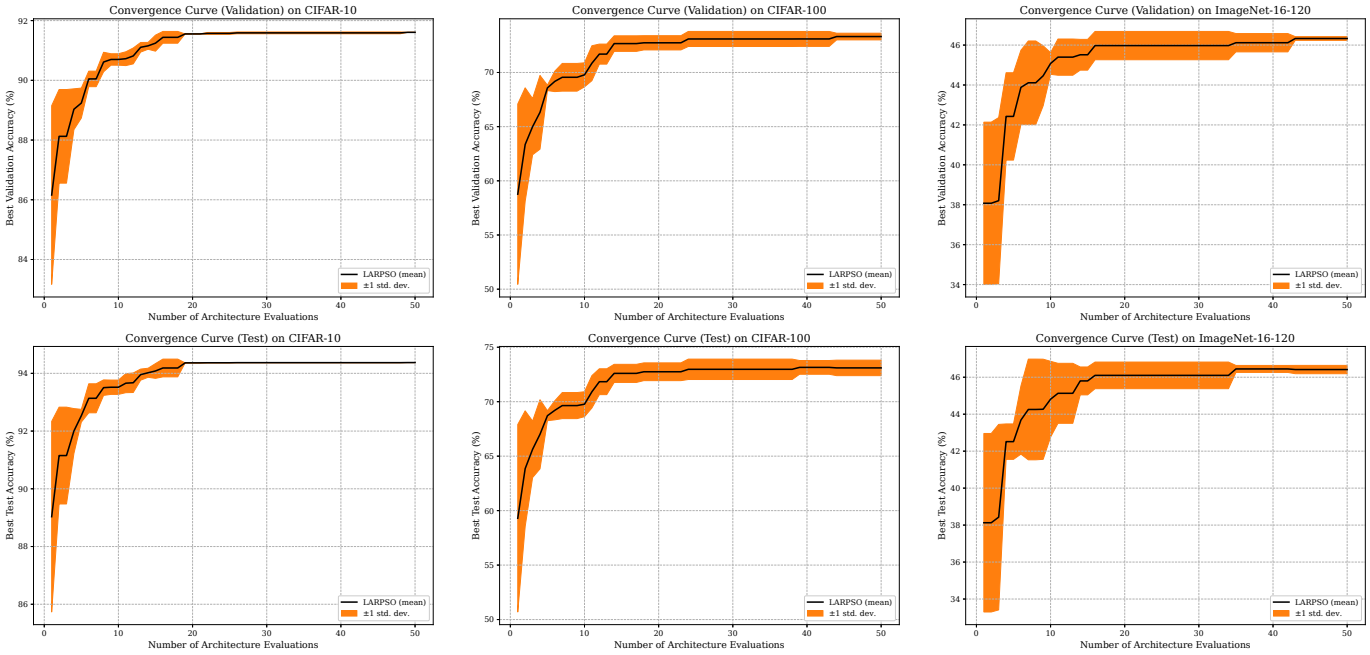

Fig. 2: Convergence curves of LARPSO on NAS-Bench-201 benchmark across CIFAR-10, CIFAR-100, and ImageNet-16-120 datasets. The top row displays the best validation accuracy found during the search process, while the bottom row shows the corresponding test accuracy. The solid black lines represent the mean performance, and the orange shaded regions indicate the standard deviation across three independent trials.

TABLE II: A qualitative example of particle evolution during a single generation in one CIFAR-10 trial.

Role	Architecture String	Val. Acc (%)
Step 1: Cognitive Learning Mutation		
Current Arch	|nor_conv_3x3~0|+|nor_conv_3x3~0|nor_conv_1x1~1|+|none~0|nor_conv_3x3~1|skip_connect~2|	89.53
Good Ref.	|nor_conv_3x3~0|+|nor_conv_3x3~0|nor_conv_1x1~1|+|none~0|nor_conv_3x3~1|skip_connect~2|	89.53
Better Ref.	|nor_conv_3x3~0|+|nor_conv_3x3~0|nor_conv_3x3~1|+|skip_connect~0|skip_connect~1|nor_conv_3x3~2|	91.12
New Arch	|nor_conv_3x3~0|+|nor_conv_3x3~0|nor_conv_3x3~1|+|skip_connect~0|nor_conv_3x3~1|skip_connect~2|	91.08
Step 2: Social Learning Mutation		
Current Arch	|nor_conv_3x3~0|+|nor_conv_3x3~0|nor_conv_3x3~1|+|skip_connect~0|nor_conv_3x3~1|skip_connect~2|	91.08
Good Ref.	|nor_conv_3x3~0|+|nor_conv_3x3~0|nor_conv_1x1~1|+|none~0|nor_conv_3x3~1|skip_connect~2|	89.53
Better Ref.	|nor_conv_3x3~0|+|nor_conv_3x3~0|nor_conv_3x3~1|+|skip_connect~0|nor_conv_3x3~1|nor_conv_3x3~2|	91.55
New Arch	|nor_conv_3x3~0|+|nor_conv_3x3~0|nor_conv_3x3~1|+|skip_connect~0|nor_conv_3x3~1|nor_conv_1x1~2|	91.61

advantage of LARPSO over these baselines becomes more pronounced on the more challenging datasets. On CIFAR-100, LARPSO attains 73.10% test accuracy, exceeding GENIUS by 2.19% and LLMatic by 1.48%. On ImageNet-16-120 dataset, LARPSO reaches 46.41% test accuracy, surpassing GENIUS by 1.45% and LLMatic by 0.54%. These comparisons suggest that while standard LLM-based methods utilize pre-trained priors effectively, they may struggle to fine-tune architectures to the specific local landscape of a complex search space. Furthermore, the standard deviations reported for LARPSO are consistently lower than those of GENIUS and LLMatic. This indicates that the proposed method not only discovers superior architectures but also maintains greater stability across independent trials.

The convergence behavior visualized in Fig. 2 further underscores the robustness of our approach. The shaded bands are wider during the initial phase which is expected due to random initialization effects. However, they become progressively narrower as the search advances. This trend indicates that different runs tend to converge toward similar high-performance levels regardless of the starting conditions. This observation is consistent with the small standard deviations reported in Table I.

The superior performance of LARPSO can be attributed to how it integrates the LLM into an explicit optimization loop. Existing methods such as GENIUS primarily provide scalar feedback that indicates whether a candidate improved, but this feedback offers limited information about the structural changes responsible for the improvement. In contrast, LARPSO prompts the LLM with paired reference architectures drawn from the swarm history that reflect a transition from a weaker solution to a stronger one. This setup encourages the LLM to infer an improvement pattern and apply an analogous transformation to the current architecture, which helps mitigate stagnation and supports more directed exploration.

When compared with other state-of-the-art methods such as

DRNAS, β-DARTS, and Λ-DARTS, we observe that LARPSO achieves optimal performance on the CIFAR-10 dataset. However, on more complex tasks involving the CIFAR-100 and ImageNet-16-120 datasets, our method exhibits slightly lower performance than these differentiable methods. Our current mutation constraint deliberately limits each update to one or two operation changes to ensure stable local refinement and prevent destructive loss of structural integrity. However, this conservative design may restrict the probability of making larger structural jumps beneficial for escaping local optima in complex optimization landscapes.

VI. Conclusion and Future Work

In this paper, we introduce LARPSO to address a key limitation of existing LLM-based NAS methods, in which mutation operations often lack explicit guidance from the evolutionary history. By integrating the learning dynamics of PSO into LLM-driven mutation, LARPSO provides the LLM with explicit improvement trajectories derived from the swarm search history. Specifically, the method prompts the LLM with paired reference architectures that reflect a transition from a weaker solution to a stronger one. This mechanism encourages the model to infer structural improvement patterns and apply analogous transformations to the current solution, thereby mitigating stagnation and promoting more directed exploration. Experiments on NAS-Bench-201 across three datasets demonstrate that LARPSO consistently outperforms representative LLM-based NAS baselines and remains competitive with state-of-the-art NAS methods.

For future work, we note that the current mutation constraint in LARPSO limits each update to one or two operation changes, which emphasizes local refinement and improves stability. This restriction may also reduce the likelihood of making larger structural changes that are beneficial for escaping local optima. Consequently, future research will focus on addressing this limitation by developing an adaptive mutation mechanism. A promising strategy is to automatically permit larger structural edits when evolutionary progress stalls.

References

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proc. Int. Conf. Neural Networks, vol. 4, 1995, pp. 1942–1948.

[2] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in Proc. Int. Conf. Learn. Represent. (ICLR), 2017.

[3] H. Kitano, "Designing neural networks using genetic algorithms with graph generation system," Complex Systems, vol. 4, pp. 461–476, 1990.

[4] H. Liu, K. Simonyan, and Y. Yang, "DARTS: differentiable architecture search," in Proc. Int. Conf. Learn. Represent. (ICLR), 2019.

[5] C. Termritthikun, Y. Jamtsho, J. Ieamsaard, P. Muneesawang, and I. Lee, "EEEA-Net: an early exit evolutionary neural architecture search," Engineering Applications of Artificial Intelligence, vol. 104, p. 104397, 2021.

[6] W. X. Zhao et al., "A survey of large language models," arXiv preprint arXiv:2303.18223, 2023.

[7] A. Chen, D. Dohan, and D. So, "EvoPrompting: language models for code-level neural architecture search," in Proc. Advances in Neural Information Processing Systems (NeurIPS), vol. 36, 2023, pp. 7787–7817.

[8] M. Zheng et al., "Can GPT-4 perform neural architecture search," arXiv preprint arXiv:2304.10970, 2023.

[9] M. U. Nasir, S. Earle, J. Togelius, S. James, and C. Cleghorn, "LLMatic: neural architecture search via large language models and quality diversity optimization," in Proc. Genetic and Evolutionary Computation Conf. (GECCO), 2024, pp. 1110–1118.

[10] X. Dai et al., "FBNetV3: joint architecture-recipe search using predictor pretraining," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2021, pp. 16276–16285.

[11] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: a survey," J. Mach. Learn. Res., 2019.

[12] M. Tan and Q. V. Le, "EfficientNet: rethinking model scaling for convolutional neural networks," in Proc. Int. Conf. Mach. Learn. (ICML), 2019.

[13] Y. Xu et al., "PC-DARTS: partial channel connections for memory-efficient architecture search," arXiv preprint arXiv:1907.05737, 2019.

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

[15] M. Zhang, S. W. Su, S. Pan, X. Chang, E. M. Abbasnejad, and R. Haffari, "iDARTS: differentiable architecture search with stochastic implicit gradients," in Proc. Int. Conf. Mach. Learn. (ICML), 2021, pp. 12557–12566.

[16] X. Dong and Y. Yang, "Searching for a robust neural architecture in four GPU hours," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2019, pp. 1761–1770.

[17] X. Chen, R. Wang, M. Cheng, X. Tang, and C. J. Hsieh, "DrNAS: Dirichlet neural architecture search," in Proc. Int. Conf. Learn. Represent. (ICLR), 2021.

[18] P. Ye, B. Li, Y. Li, T. Chen, J. Fan, and W. Ouyang, "B-DARTS: beta-decay regularization for differentiable architecture search," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2022, pp. 10874–10883.

[19] S. Movahedi et al., "λ-DARTS: mitigating performance collapse by harmonizing operation selection among cells," in Proc. Int. Conf. Learn. Represent. (ICLR), 2023.

[20] K. T. Chitty-Venkata, M. Emani, V. Vishwanath, and A. K. Somani, "Neural architecture search benchmarks: insights and survey," IEEE Access, vol. 11, pp. 25217–25236, 2023.

[21] X. Dong and Y. Yang, "NAS-Bench-201: extending the scope of reproducible neural architecture search," arXiv preprint arXiv:2001.00326, 2020.

[22] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. of Toronto, Tech. Rep., 2009.

[23] P. Chrabaszcz, I. Loshchilov, and F. Hutter, "A downsampled variant of ImageNet as an alternative to the CIFAR datasets," arXiv preprint arXiv:1707.08819, 2017.

[24] L. Xie et al., "Weight-sharing neural architecture search: a battle to shrink the optimization gap," ACM Computing Surveys, vol. 54, no. 9, pp. 1–37, 2021.

[25] P. Ren et al., "A comprehensive survey of neural architecture search: challenges and solutions," ACM Computing Surveys, vol. 54, no. 4, pp. 1–34, 2021.

[26] J.-B. Mouret and J. Clune, "Illuminating search spaces by mapping elites," arXiv preprint arXiv:1504.04909, 2015.

[27] Y. Shinohara, J. Xu, T. Li, and H. Iba, "Large language models as particle swarm optimizers," arXiv preprint arXiv:2504.09247, 2025.

[28] R. Matai, S. P. Singh, and M. L. Mittal, "Traveling salesman problem: an overview of applications, formulations, and solution approaches," Traveling Salesman Problem, Theory and Applications, vol. 1, no. 1, pp. 1–25, 2010.

[29] J. K. Pugh, L. B. Soros, and K. O. Stanley, "Quality diversity: a new frontier for evolutionary computation," Front. Robot. AI, vol. 3, p. 40, July 2016.