A Task-Aware Rank-Based Trained LLM Surrogate for Cross-Task Neural Architecture Search

Kai-Chun Wu, Hao-Qun Huang, Hsin-Yu Wang, and Chun-Wei Tsai
Department of Computer Science and Engineering
National Sun Yat-Sen University
Kaohsiung, Taiwan
{kaijinkaijin2, alvinhuang0603, angel.ilc.xinmin}@gmail.com, cwtsai@cse.nsysu.edu.tw

***Abstract*—Using large language models (LLMs) to improve generalization across unseen out-of-domain tasks is promising in neural architecture search (NAS), yet it still struggles to solve regression problems precisely when performance distributions shift across domains. To address the issue, the proposed framework strategically integrates several existing methods: (1) employing rank-based contrastive loss, which focuses on learning-to-rank by the candidates relative comparisons, (2) take advantage of hierarchical pairwise sampling to emphasize fine-grained distinctions among top-performing architectures but still maintain a global ordering and (3) utilizing decoder-based large language models to enable potential inference ability gains. To verify the performance of the framework, it is evaluated on the NAS-Bench-201 and TransNAS-Bench-101 benchmarks. Experimental results provide strong evidence that the proposed strategic integration enables LLMs to adapt to regression problems and demonstrates significantly more effective transferability, achieving superior zero-shot performance compared to conventional strong predictors.**

***Index Terms*—Surrogate-assisted data-driven optimization, large language models, and transfer optimization**

I. Introduction

Neural architecture search (NAS) aims to automate the manual design process of neural networks, thereby liberating researchers from trial-and-error adjustments for traditional NAS. Candidate architectures are often trained from scratch [1], where the evaluation stage often accounts for most of the total computation cost. [1], [2] To overcome this bottleneck, researchers have investigated a wide range of proxy approaches, including statistical proxies (e.g. zero-cost proxies [3], performance predictors [1]), machine learning surrogates (e.g. bayesian optimization [1], random forests [1], [4]), and deep learning-based predictors [5]. These surrogate models are trained on a limited set of fully evaluated architectures, enabling them to estimate the performance of new candidates without exhaustive training. As fundamental research on NAS predictor models approaches saturation, recent developments have increasingly shifted toward cross-task generalization, where a model trained on one or more source tasks is expected to effectively transfer to target out-of-domain (OOD) tasks with different target distributions [4], [6]. Motivated by this progress, NAS predictors have gradually transitioned from traditional machine learning approaches to deep learning–based models, which demonstrate stronger representational capacity and transferability [1], [5].

Existing LLM-based NAS surrogate models [4], [6] typically employ regression objectives (e.g., MSEs) to predict absolute performance values. However, in cross-task settings, performance metrics can vary significantly across datasets (e.g., accuracy as metric in dataset `CIFAR-10` versus complex tasks such as dataset `AutoEncoder` uses SSIM). This scale mismatch complicates the surrogate's ability to generalize across new domains. In contrast, learning-to-rank (LTR) approaches suggest that focusing on relative comparisons between architectures provides a more stable and transferable signal across tasks [7], [8]. Prior studies in NAS have demonstrated that pairwise ranking losses enhance the correlation between predicted and actual rankings compared to point-wise regression [9], [10]. Motivated by these insights, we conclude that emphasizing relative ordering over absolute performance values enables LLMs to better capture architectural patterns that generalize across diverse tasks.

To tackle these challenges, we propose the task-aware rank-based trained LLM surrogate (RLMS). The proposed framework enables effective zero-shot cross-task transfer by integrating the following components:

1) Hierarchical pairwise sampling: It constructs training pairs at multiple difficulty levels [11], facilitating the learning process with rank-based contrastive loss which is crucial for effective ranking in cross-task scenarios.
2) Rank-based contrastive loss [7], [8]: This approach enables the model to learn relative preferences with both global ordering and fine-grained distinctions among top-performing architectures.
3) Decoder-based LLM backbone: We utilize decoder-based LLMs (e.g., Gemma–2B, LLaMA–2–2B, and Qwen–2B [12]–[14]) as the surrogate backbone, leveraging their strong language generation capabilities and flexibility in processing diverse architecture representations and task descriptions.

In Section II, we review the foundational works and prior research that have enabled our contributions. The proposed framework is introduced in Section III, which explains how we perform generalization across tasks and the methodology behind it. Section IV demonstrates experiments and illustrates our results. Finally, in Section V, we discuss the conclusion and the future work.

II. Related Works

Neural architecture search (NAS) can be formulated as an optimization problem over a discrete architecture space. Let

$\mathcal{A}$ denote the search space of candidate architectures, and let $\mathcal{T}$ denote a learning task characterized by a dataset, training setup, and evaluation metric. For a given task $\mathcal{T}$, the objective of NAS is to identify

$$a^*_{\mathcal{T}} = \arg\max_{a \in \mathcal{A}} \mathcal{F}(a, \mathcal{T}), \tag{1}$$

where $\mathcal{F}(a, \mathcal{T})$ denotes the true performance of architecture a on task $\mathcal{T}$. In practice, directly evaluating $\mathcal{F}(a, \mathcal{T})$ requires training the architecture from scratch, which is computationally expensive. As a result, NAS is commonly decomposed into three components: (i) a search space that defines how architectures are parameterized, (ii) a search strategy that explores the space, and (iii) an evaluation strategy that estimates architecture performance.Among these components, the evaluation stage typically dominates the overall computational cost, motivating the use of surrogate models, weight-sharing schemes, or other proxy evaluations to efficiently approximate $\mathcal{F}(a, \mathcal{T})$.

Most existing NAS methods operate under a *single-task* assumption, where surrogate models are trained and applied on the same task $\mathcal{T}$. In contrast, *cross-task NAS* considers a more general setting in which knowledge acquired from multiple source tasks is transferred to guide architecture search on an unseen target task [15]. Formally, let $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_K\}$ denote a set of source tasks, and let $\mathcal{T}_{\text{target}}$ denote a target task that is unavailable during surrogate training. The goal of cross-task NAS is to leverage observations from the source tasks to accelerate architecture evaluation and search on $\mathcal{T}_{\text{target}}$ without extensive task-specific retraining [16]. A fundamental challenge in this setting is that the performance function $\mathcal{F}(a, \mathcal{T})$ is inherently task-dependent and not directly comparable across tasks. Differences in data distributions, task difficulty, and evaluation metrics often lead to substantial shifts in performance scales, making it difficult for a single surrogate model to generalize across tasks when predicting absolute performance values.

A. Evaluation Strategies for Architecture Search

To reduce the immense computational cost of NAS, performance predictors have been proposed to estimate architecture performance without full training [1]. Early approaches employ graph-based neural networks or Bayesian optimization methods, such as BANANAS [17], to learn mappings from encoded architectures to their expected accuracy [1], [5]. These methods achieve strong performance within a fixed search space, where the architectures, dataset, and training pipeline remain unchanged [5]. Concurrently, zero-cost proxies (ZCPs) have emerged as training-free alternatives [3]. Instead of performing full training, ZCPs such as SynFlow [18] and GradNorm compute lightweight statistics from one or a few mini-batches at initialization and use them to rank candidate architectures [1], [19]. This design enables the evaluation of thousands of architectures using only a limited number of forward or backward passes [1]. Despite their efficiency, both predictors and zero-cost proxies suffer from limited robustness across tasks. Large-scale studies show that a proxy effective on one benchmark (e.g., NAS-Bench-201) may produce near-random rankings on another (e.g., NAS-Bench-101) [4], [5]. Similarly, neural-network-based predictors are typically trained on a single dataset with a fixed architecture encoding, leading to overfitting to the original search space and degraded performance when applied to new tasks or heterogeneous settings [4], [5].

B. Large Language Models for Architecture Search

The emergence of large language models (LLMs) has unlock direct access and generation of code, text, and symbolic descriptions, enabling more flexible representations of the search space. Early LLM-based studies, such as GENIUS and LLMatic, primarily employed LLMs as assistants within existing search pipelines [19]–[21]. In these approaches, LLMs are used to propose mutation operators, suggest new architectures that is subsequently evaluated by a standard training loop.

More recent work has explored the use of LLMs directly as surrogate models [22]. LLM-based surrogates leverage flexible textual descriptions of architectures and thus operate over more expressive and variable search spaces [4]. For instance, Zhang et al. propose a meta-surrogate framework that embeds task metadata into the prompt, allowing a single model to generalize across different optimization contexts [6]. Similarly, Qin et al. demonstrate that fine-tuning pre-trained language models on architecture strings (e.g., BERT or LLaMA style models) yields predictors that generalize across datasets rather than being restricted to a single benchmark [4]. Overall, these approaches alleviate the input flexibility limitations of traditional deep learning predictors and provide a more adaptable interface for cross-task NAS.

However, despite these advances, robust cross-task generalization remains an open challenge. Existing LLM-based surrogates predominantly rely on point-wise regression objectives, which are inherently sensitive to task-specific performance scales and evaluation regimes [4], [6]. Empirical evidence suggests that such objectives struggle to preserve meaningful ranking information when transferred across heterogeneous tasks, leading to degraded search performance [4]. In contrast, learning-to-rank methods offer a complementary perspective by emphasizing relative architectural comparisons, which have been shown to provide more stable and transferable signals across tasks [7]–[9]. Nevertheless, the integration of ranking-based objectives with LLM-based surrogate models has not been fully explored. This gap motivates our work, which seeks to combine the representational flexibility of LLMs with ranking-oriented learning objectives to enable effective and scalable cross-task neural architecture search.

III. Proposed Method

A. Overview of the Cross-Task LLM Surrogate Framework

In this study, we propose the task-aware rank-based trained LLM surrogate (RLMS) for cross-task neural architecture search (NAS), which uses an LLM as a text-based surrogate to score architectures under a given task condition and is trained to preserve relative preferences rather than regress absolute performance values. Figure 1 summarizes the framework, which

consists of two components: an *LLM regressor* and a *rank contrastive loss function*.

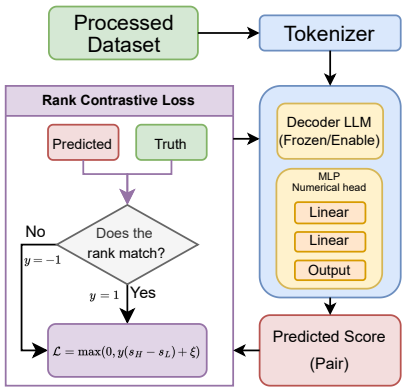

Fig. 1. Overview of RLMS. The framework employs a decoder-only LLM and connects its hidden representations to a multi-layer perceptron to form an integrated surrogate model. A margin ranking loss is used to implement the learning-to-rank objective. The preprocessed dataset is typically drawn from a source set (e.g., NAS-Bench-201).

Let $\mathcal{A}$ denote the architecture search space and let $\tau \in \mathcal{T}$ denote a task. For each candidate architecture $a \in \mathcal{A}$, we form a textual input by concatenating a task descriptor d_τ (derived from task metadata) and an architecture string s_a (serialized from the topology). An LLM-based surrogate f_θ maps this input to a scalar score:

$$s(a, \tau) = f_\theta([d_\tau; s_a]). \tag{2}$$

During *source-task tuning*, RLMS is trained using within-task pairwise comparisons [7]. Given two architectures (a_i, a_j) sampled from the same source task, the model is optimized to assign a higher score to the better-performing architecture. This design reduces sensitivity to task-specific score scales and encourages the surrogate to focus on relative quality. During *target-task zero-shot scoring*, the tuned surrogate is directly applied to an unseen target task by conditioning on $d_{\tau_{\text{target}}}$. The predicted scores are then used to rank candidate architectures and guide the downstream search procedure (e.g., evolutionary search).

B. Architecture and Task Representation

RLMS uses a shared textual format to describe both the task and the candidate architecture. For each task $\tau \in \mathcal{T}$, we construct a task description d_τ from available metadata. For each architecture $a \in \mathcal{A}$, we convert its topology into an architecture string s_a using a fixed set of grammar rules. The surrogate then scores the combined input $[d_\tau; s_a]$.

1) Architecture String Format: Cell-based search spaces such as NAS-Bench-201 [23] represent architectures as Directed Acyclic Graphs (DAGs). When using LLMs as surrogates, a key challenge is converting this non-sequential graph structure into a linear token sequence without losing essential connectivity information. To handle this, we adopt a topology-aware token format that encodes both the operation at each node and its incoming connection. Each node in the DAG is serialized as a structured token of the form `(op: name, in: input_index)`. This separates architectural information into two parts: Operation type (`op`) specifies the computation (e.g., `nor_conv_3x3`), capturing primitive-level details; and Input reference (`in`) indicates the predecessor node index, explicitly encoding dependency and information flow.

An architecture a is thus represented as a sequence $s_a = [t_1, t_2, \ldots, t_N]$, where each token t_i corresponds to one node and its incoming edge. By preserving both operation semantics and connectivity, this representation provides a faithful linear description of the original cell structure that can be directly processed by the LLM.

2) Task Description: For cross-task transfer, we attach a task description d_τ to each input. Rather than using a learned task embedding, d_τ is written as a structured description derived from task metadata. It includes the following fields: Domain and dataset (e.g., "image classification on CIFAR-100"); data characteristics covering input size and label granularity; target metric (e.g., "Top-1 Accuracy"); and hints for optional domain guidance. This description provides task context during scoring. For example, heavy downsampling may be less suitable for pixel-wise prediction tasks due to the loss of spatial detail, but can be effective for global recognition tasks.

3) Model Input: We form the model input by concatenating the task description and the architecture string with explicit separators:

$$x(a, \tau) = \texttt{[Task]}\ d_\tau\ \texttt{[Arch]}\ s_a. \tag{3}$$

This format follows the standard input style of decoder-based language models and allows the surrogate to output a scalar score conditioned on both the task and the architecture.

C. Rank-Based Training and Zero-Shot Scoring

To improve cross-task transfer under varying performance scales, RLMS replaces point-wise regression with a pairwise ranking formulation.

1) Model Architecture: We implement the surrogate f_θ with a decoder-only LLM (e.g., Gemma-2B) [12]. To fully adapt the pre-trained representations to the NAS domain, we perform full-parameter fine-tuning on the LLM backbone. Given the input sequence $x = [d_\tau; s_a]$, we use the hidden state of the last token (EOS) $h_{\text{eos}} \in \mathbb{R}^D$ as the representation. A two-layer MLP head then maps h_{eos} to a scalar score:

$$s(a, \tau) = \mathbf{w}_2^\top \sigma(\mathbf{W}_1 h_{\text{eos}} + \mathbf{b}_1) + b_2, \tag{4}$$

where σ is ReLU and $\{\mathbf{W}_1, \mathbf{b}_1, \mathbf{w}_2, b_2\}$ are trainable parameters.

2) Hierarchical Pairwise Sampling: Uniform random sampling for ranking-based training often produces pairs with large performance gaps, which provide limited supervision for fine-grained ordering. Motivated by the idea of hard negative mining [11], we adopt a hierarchical pairwise sampling scheme covering three levels: (1) *Top-tier hard pairs* (top-25%) focus on fine-grained discrimination among strong candidates, which is critical for the later stages of search; (2) *Mid-tier hard pairs* (25%–75%) are sampled to maintain smooth ordering and prevent large ranking gaps; and (3) *Coarse-level pairs* (top/bottom-50%) provide clear global ordering signals to stabilize training.

During training, pairs from both levels are mixed to balance global ordering and local refinement. For each sampled pair, we apply a margin ranking loss:

$$\mathcal{L} = \max(0,\ \xi - (s(a_i) - s(a_j)))\,, \tag{5}$$

where $s(\cdot)$ denotes the score predicted by the surrogate and ξ is the margin (e.g., $\xi = 0.2$). This sampling scheme allows the model to learn a consistent global ranking while gradually improving discrimination among top-performing architectures.

3) Zero-Shot Scoring and Efficiency: At search time, we apply the trained surrogate to an unseen target task τ_{tgt} without further tuning. Instead of comparing all pairs, we score each candidate once and rank them by the predicted scores:

$$\mathcal{S}(\tau_{\text{tgt}}) = \{\, s(a, \tau_{\text{tgt}}) \mid a \in \mathcal{A}_{\text{cand}} \,\}. \tag{6}$$

This yields $O(N)$ scoring complexity for N candidates, which is directly applicable to evolutionary search or other ranking-based selection loops.

IV. Experiment and Analysis

A. Environment Settings and Evaluation Metrics

All of the experiments are performed on two distinct hardware configurations: (1) Laboratory Workstation: An Intel Xeon W7-3465X workstation (128GB RAM) equipped with four NVIDIA A6000 GPUs (48GB each). [1] (2)Personal Workstation: An Intel Core i9-9820X with 2 GPUs (NVIDIA RTX 2080ti, 32GB RAM). [2]

For reproducibility, all experiments are conducted with fixed random seeds, and the reported results are averaged over multiple runs. All methods within a given experiment are executed on the same hardware unless otherwise specified. [3]

Following the standard evaluation protocol in Neural Architecture Search, we adopt Spearman's ρ and Kendall's τ rank correlation coefficients [24], [25] to assess global monotonic agreement and pairwise ordering consistency under distribution shifts, respectively. To conduct a rigorous and reproducible evaluation, we benchmark RLMS against strong traditional machine learning methods from three complementary categories. Following the recommendations of White et al. [1], we select these baselines because large-scale empirical studies consistently show that well-tuned traditional models often match or exceed the accuracy and stability of highly specialized deep learning surrogates.

- Tree-based methods: We include Random Forest (RF), XGBoost, and LightGBM. These ensemble models consistently achieve competitive performance and serve as strong traditional predictors when architectures are represented in a tabular or feature-based form [1].
- Graph-based methods: We employ Graph Convolutional Networks (GCN) [26], which explicitly model the graph structure of neural architectures and capture topological dependencies between operations.
- Bayesian and statistical methods: We include DNGO [27], which applies Bayesian linear regression for uncertainty-aware prediction, and Synflow [18], a representative zero-cost proxy that estimates architecture quality without training.

All traditional methods are implemented using the official NASLib framework [1] with the recommended hyperparameter settings. This setup ensures that our comparisons are made against well-tuned implementations and reflect the practical upper-bound performance of established predictors.

TABLE I
RANKING CORRELATION AND SEARCH PERFORMANCE ON NAS-BENCH-201

Target	Method	τ	ρ	Best Rank	Avg. Rank
CIFAR-10	RF	0.5974±0.001	0.7894±0.001	53	788.6
	GCN	0.6698±0.014	0.8553±0.011	11	318.8
	RLMS	**0.7990±0.028**	**0.9422±0.017**	**6**	**10.0**
CIFAR-100	RF	0.6097±0.001	0.7972±0.001	46	149.2
	GCN	0.6235±0.008	0.8158±0.007	6	169.8
	RLMS	**0.7948±0.025**	**0.9405±0.037**	**1**	**1.0**
ImageNet-16-120	RF	0.6083±0.002	0.7972±0.002	455	1904.4
	GCN	0.6284±0.011	0.8211±0.010	1482	2357.2
	RLMS	**0.7217±0.023**	**0.8887±0.018**	**1**	**1.0**

TABLE II
OVERLAP RATIO BETWEEN PREDICTED AND GROUND-TRUTH TOP-K ARCHITECTURES ON NAS-BENCH-201

Target	Method	@50	@100	@500
CIFAR-10	RF	4.4±0.8	11.6±0.5	26.2±1.2
	GCN	10.4±8.3	18.0±8.2	35.6±5.3
	RLMS	**38.8±4.6**	**42.6±1.5**	**66.4±1.2**
CIFAR-100	RF	22.4±0.8	30.8±1.2	61.8±0.1
	GCN	15.6±7.1	20.8±5.5	39.2±3.6
	RLMS	**51.2±3.9**	**49.6±3.9**	**60.6±2.1**
ImageNet-16-120	RF	9.6±0.8	10.0±0.0	29.8±0.2
	GCN	0.4±0.8	0.8±1.6	14.4±4.6
	RLMS	**64.8±16.0**	**57.4±10.0**	**55.4±6.9**

Overlap@K measures the intersection between predicted top-K architectures and ground truth.

B. Within-Task Evaluation on NAS-Bench-201

We first evaluate RLMS within NAS-Bench-201 [23] using leave-one-out cross-validation. We compare it with Random Forest (RF) and GCN to verify its fundamental ranking behavior before examining cross-task transferability. In terms of Table I, RLMS yields consistently stronger ranking correlation achieving Kendall's τ of 0.7990, substantially higher than GCN (0.6698) and RF (0.5974).

While RF and GCN occasionally identify competitive candidates, their performance degrades noticeably on more challenging settings. On ImageNet-16-120, in particular, GCN exhibits a severe drop in effectiveness, with an average rank of 2357.2, suggesting limited robustness to increased task difficulty. In contrast, RLMS consistently places the optimal architecture at the top, maintaining an average rank of 1.0.

[1]Laboratory Workstation: Large models such as Gemma-2B is trained with this setup. In each training run, two GPUs are employed using data parallelism, while the remaining GPUs allow concurrent execution of other runs.

[2]Personal Workstation: This setup is used for experiment involving smaller models in the experiment such as Gemma-3-270m-it.

[3]We employ the AdamW optimizer with a learning rate of 5×10^{-4}, a batch size of 32, and a cosine annealing scheduler for 6 epochs. The max sequence length is set to 512, and the ranking margin ξ is fixed at 0.2. For LoRA fine-tuning, we use a rank $r = 16$ and alpha $\alpha = 32$. All results are averaged over 5 fixed random seeds.

Beyond individual best results, RLMS also shows a clear advantage in prioritizing strong candidates overall. The higher overlap ratios at @50 and @100 on Table II indicate that architectures ranked highly by RLMS are much more likely to belong to the true top-performing set, which is a critical property for practical search algorithms that rely on early-stage candidate selection.

C. Zero-Shot Cross-Task Transfer

We evaluate the cross-task generalization of RLMS through zero-shot transfer experiments on the TransNAS-Bench-101 [16] benchmark. This dataset is specifically constructed to assess the transferability of NAS methods across diverse domains. Crucially, we utilize its micro search space, which consists of 4,096 architectures derived from the same topological structure (5 operations, 4 nodes) as our source domain, NAS-Bench-201 [23]. This structural alignment ensures that the architecture encodings learned on the source task are compatible with the target tasks. In this setting, the surrogate is directly applied to rank architectures on seven unseen target tasks without any gradient updates or task-specific fine-tuning. The comparative results against the full set of traditional methods are summarized in Figure 2. RLMS performs strongly across most target tasks, ranking first in 6 out of 7 cases. To provide a concise summary of its overall behavior, we report the average performance across all tasks in Table III. RLMS attains an average Kendall's τ of 0.505, exceeding the strongest traditional method, DNGO (0.488), and outperforming graph-based predictors such as GCN (0.464). These results indicate that the proposed rank-based LLM surrogate transfers architectural knowledge effectively, even when applied to tasks that differ substantially from the training domain.

TABLE III
AVERAGE ZERO-SHOT TRANSFER PERFORMANCE ACROSS 7 TRANSNAS-BENCH-101 TASKS

Method	τ	ρ
Random Forest	0.407	0.566
GCN	0.464	0.639
XGBoost	0.402	0.566
MLP	0.403	0.562
Synflow	0.363	0.513
DNGO	0.488	0.653
LightGBM	0.402	0.560
RLMS (Ours)	**0.505**	**0.684**

Among all the traditional predictors, DNGO (Bayesian Linear Regression) emerges as the most competitive alternative, achieving the second-best average performance and slightly surpassing RLMS on the `class_scene` task (τ: 0.687 vs. 0.675). This behavior aligns with prior findings [1] that Bayesian methods are particularly effective in zero-shot settings, where uncertainty estimation helps prevent overfitting. Notably, `class_scene` shares the same task paradigm as the source task (image classification) and exhibits similar visual characteristics. In this case, the transfer mainly involves a shift in data distribution rather than a change in the learning objective, allowing DNGO's kernel-based modeling to remain effective.

The limitations of traditional predictors become more apparent on heterogeneous tasks where the learning objective itself changes. For tasks such as `room_layout` (regression) and `jigsaw` (self-supervised learning), methods such as GCN and DNGO exhibit noticeable performance degradation. Because these predictors rely solely on architecture topology encodings, which remain unchanged across tasks, they are unable to account for shifts in what constitutes a suitable architecture.

In contrast, RLMS explicitly conditions its ranking on the task description. By incorporating semantic information about the target objective (e.g., regression versus classification), the LLM adjusts its evaluation criteria accordingly. As a result, RLMS achieves a Kendall's τ of 0.402 on `room_layout`, outperforming DNGO (0.369). This comparison suggests that while statistical predictors are robust to *data distribution shifts*, RLMS is better suited to handle *task paradigm shifts*, which are common in practical cross-task NAS scenarios.

It is worth noticing that all reported results are obtained using a relatively compact Gemma-2B backbone. This indicates that the performance gains of RLMS do not primarily arise from model scale, but from the combination of rank-based training and explicit task conditioning. Meanwhile, since these mechanisms are largely model-agnostic, scaling the backbone to larger language models (e.g., LLaMA-7B) is expected to further enhance performance, suggesting additional potential beyond the current setting.

V. CONCLUSIONS

In this study, we propose RLMS, a novel framework that employs large language models (LLMs) as rank-based surrogate predictors for cross-task neural architecture search (NAS). Unlike conventional regression-based predictors, RLMS is trained to learn relative architectural preferences via pairwise comparisons under explicit task conditioning, thereby mitigating the sensitivity to task-specific performance scales that often undermines cross-domain generalization. By integrating structured task descriptions with a hierarchical pairwise sampling strategy, the proposed method effectively captures transferable architectural patterns and enables robust zero-shot transfer to unseen target tasks. Extensive experiments on NAS-Bench-201 and TransNAS-Bench-101 demonstrate that RLMS consistently outperforms a diverse set of traditional predictors, including tree-based ensembles, graph neural networks, and Bayesian methods. Notably, RLMS exhibits strong robustness under task paradigm shifts, where the learning objective itself changes across domains. Comprehensive ablation studies further validate the effectiveness of the ranking-based training objective and confirm that RLMS maintains stable performance across a wide range of hyperparameter configurations. Our few-shot adaptation experiments in high-shift scenario can efficiently exploit source-domain priors with only minimal target-task supervision. Enabling a new promising direction for leveraging LLMs in NAS, RLMS opens up exciting avenues for future

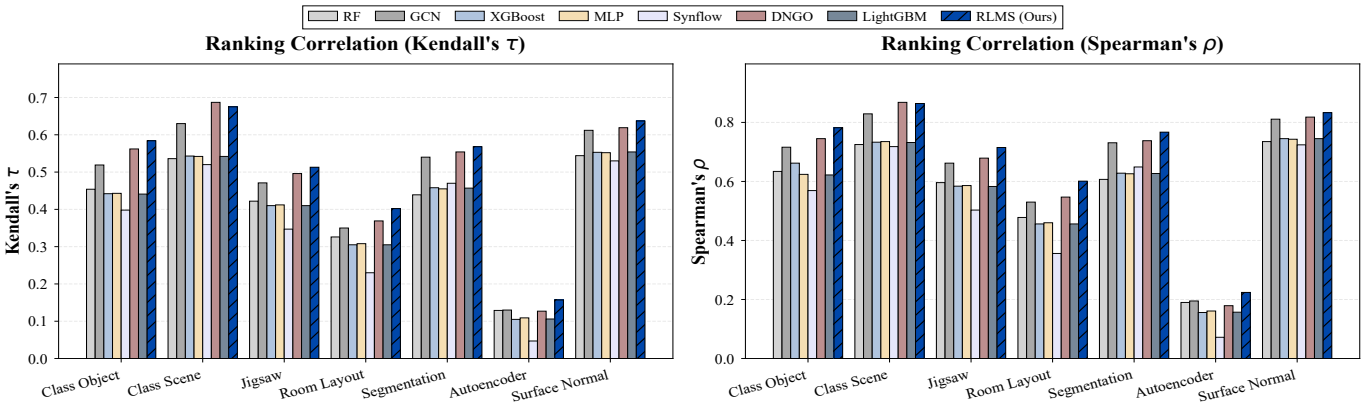

Fig. 2. Zero-shot ranking correlation on TransNAS-Bench-101. RLMS consistently achieves higher correlations than traditional methods across diverse tasks.

research on scalable and generalizable architecture optimization and even automate the design of LLMs themselves.

ACKNOWLEDGMENTS

This work was supported in part by the National Science and Technology Council of Taiwan, R.O.C., under Contract NSTC112-2628-E-110-001-MY3, NSTC114-2634-F-110-001-MBK, and NSTC114-2221-E-110-023-MY3. During the preparation of this manuscript, the authors utilized Gemini model family in order to improve English readability and language phrasing. After using this tool, the authors carefully reviewed and edited the manuscript, and take full and complete responsibility for the final content of the publication.

REFERENCES

[1] C. White, A. Zela, B. Ru, Y. Liu, and F. Hutter, "How powerful are performance predictors in neural architecture search?" in *Proceedings of Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 28 454–28 469.

[2] W. Wen, H. Liu, Y. Chen, H. Li, G. Bender, and P.-J. Kindermans, "Neural predictor for neural architecture search," in *Proceedings of European Conference on Computer Vision*, 2020, pp. 660–676.

[3] J. Mellor, J. Turner, A. Storkey, and E. J. Crowley, "Neural architecture search without training," in *Proceedings of International Conference on Machine Learning*, 2021, pp. 7588–7598.

[4] S. Qin, G. Kadlecová, M. Pilát, S. B. Cohen, R. Neruda, E. J. Crowley, J. Lukasik, and L. Ericsson, "Transferrable Surrogates in Expressive Neural Architecture Search Spaces," in *Proceedings of International Conference on Automated Machine Learning*, 2025.

[5] F. X. Han, K. G. Mills, F. Chudak, P. Riahi, M. Salameh, J. Zhang, W. Lu, S. Jui, and D. Niu, "A General-Purpose Transferable Predictor for Neural Architecture Search," in *Proceedings of SIAM International Conference on Data Mining*, 2023, pp. 721 – 729.

[6] X.-R. Zhang, Y.-J. Gong, Y.-T. Zhong, T. Huang, and J. Zhang, "Large language model as meta-surrogate for offline data-driven many-task optimization: A proof-of-principle study," *Information Sciences*, vol. 726, p. 122762, 2026.

[7] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *Proceedings of International Conference on Machine Learning*, 2005, pp. 89–96.

[8] Y. Xu, Y. Wang, K. Han, Y. Tang, S. Jui, C. Xu, and C. Xu, "ReNAS: Relativistic Evaluation of Neural Architecture Search," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4411–4420.

[9] H. Ji, Y. Feng, J. Fan, and Y. Sun, "Loss Functions for Predictor-based Neural Architecture Search," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025, pp. 1624–1633.

[10] X. Ning, Y. Zheng, T. Zhao, Y. Wang, and H. Yang, "A Generic Graph-based Neural Architecture Encoding Scheme for Predictor-based NAS," in *Proceedings of European Conference on Computer Vision*, 2020, pp. 189–204.

[11] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 815–823.

[12] Gemma-Team, "Gemma: Open Models Based on Gemini Research and Technology," *arXiv preprint*, 2024.

[13] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "LLaMA: Open and Efficient Foundation Language Models," *arXiv preprint*, 2023.

[14] Qwen3-Team, "Qwen3 Technical Report," Alibaba DAMO Academy, Tech. Rep., 2025.

[15] C. Wong, N. Houlsby, Y. Lu, and A. Gesmundo, "Transfer learning with neural AutoML," in *Proceedings of Advances in Neural Information Processing Systems*, vol. 31, 2018, pp. 8366–8375.

[16] Y. Duan, X. Chen, H. Xu, Z. Chen, X. Liang, T. Zhang, and Z. Li, "TransNAS-Bench-101: Improving Transferability and Generalizability of Cross-Task Neural Architecture Search," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2021, pp. 5251–5260.

[17] C. White, W. Nado, O. a. Hunt, and M. Sewak, "BANANAS: Bayesian Optimization with Neural Architectures for Neural Architecture Search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 10 293–10 301.

[18] H. Tanaka, D. Kunin, D. L. Yamins, and S. Ganguli, "Pruning Neural Networks Without Any Data by Iteratively Conserving Synaptic Flow," in *Proceedings of Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 6377–6389.

[19] Z. Ji, G. Zhu, C. Yuan, and Y. Huang, "RZ-NAS: Enhancing LLM-guided Neural Architecture Search via Reflective Zero-Cost Strategy," in *Proceedings of International Conference on Machine Learning*, 2025.

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

[21] M. U. Nasir, S. Earle, J. Togelius, S. James, and C. Cleghorn, "LLMatic: Neural Architecture Search via Large Language Models and Quality-Diversity Optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2024, pp. 1110–1118.

[22] G. Jawahar, M. Abdul-Mageed, L. V. Lakshmanan, and D. Ding, "LLM Performance Predictors are good initializers for Architecture Search," in *Proceedings of the Association for Computational Linguistics*, 2024, pp. 10 540–10 560.

[23] X. Dong and Y. Yang, "NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search," in *Proceedings of International Conference on Learning Representations*, 2020.

[24] C. Spearman, "The proof and measurement of association between two things," *The American Journal of Psychology*, vol. 15, pp. 72–101, 1904.

[25] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, pp. 81–93, 1938.

[26] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *Proceedings of International Conference on Learning Representations*, 2017.

[27] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, M. Prabhat, and R. Adams, "Scalable Bayesian Optimization Using Deep Neural Networks," in *Proceedings of International Conference on Machine Learning*, vol. 37, 2015, pp. 2171–2180.