LM-Searcher: Cross-domain Neural Architecture Search with LLMs via Unified Numerical Encoding

Yuxuan Hu[1], Jihao Liu[1], Ke Wang[1], Jinliang Zheng[3,4], Weikang Shi[1], Manyuan Zhang[1], Qi Dou[2], Rui Liu[1], Aojun Zhou[1]✉, Hongsheng Li[1,4,5]✉
[1]CUHK MMLab [2]CUHK CURI [3]Tsinghua University
[4]Shanghai AI Laboratory [5]CPII under InnoHK
✉Corresponding author
{huyuxuan621,aojunzhou}@gmail.com hsli@ee.cuhk.edu.hk

Abstract

Recent progress in Large Language Models (LLMs) has opened new avenues for solving complex optimization problems, including Neural Architecture Search (NAS). However, existing LLM-driven NAS approaches rely heavily on prompt engineering and domain-specific tuning, limiting their practicality and scalability across diverse tasks. In this work, we propose LM-Searcher, a novel framework that leverages LLMs for cross-domain neural architecture optimization without the need for extensive domain-specific adaptation. Central to our approach is NCode, a universal numerical string representation for neural architectures, which enables cross-domain architecture encoding and search. We also reformulate the NAS problem as a ranking task, training LLMs to select high-performing architectures from candidate pools using instruction-tuning samples derived from a novel pruning-based subspace sampling strategy. Our curated dataset, encompassing a wide range of architecture-performance pairs, encourages robust and transferable learning. Comprehensive experiments demonstrate that LM-Searcher achieves competitive performance in both in-domain (e.g., CNNs for image classification) and out-of-domain (e.g., LoRA configurations for segmentation and generation) tasks, establishing a new paradigm for flexible and generalizable LLM-based architecture search. The datasets and models will be released at https://github.com/Ashone3/LM-Searcher.

1 Introduction

Recent advances in Large Language Models (LLMs) have demonstrated remarkable potential in solving complex optimization problems, including competitive programming (OpenAI, 2024), the Traveling Salesman Problem (TSP) (Yang et al., 2023), and even Neural Architecture Search (NAS) (Zheng et al., 2023; Nasir et al., 2024). Among these applications, NAS focuses on discovering optimal neural architectures that maximize performance in a given domain. Previous approaches, such as GENIUS and LLMatic, leverage GPT-4 (OpenAI, 2023) to generate task-specific architectures, either in code or natural language form.

These preliminary attempts focus on designing prompts to instruct off-the-shelf LLMs for the architectural design of image classification models. However, prompt engineering-based approaches require extensive domain-specific expertise and manual prompt tuning to handle different search spaces and tasks, making them less practical in real-world scenarios. For instance, while GPT-4o can effectively identify better architectures within a simple classification search space such as CIFAR-10, it fails to surpass the random search approach when applied to larger search spaces like Diffusion Transformer (Peebles and Xie, 2023) for image generation.

In this paper, we ask the following question: *Can LLMs be trained as general-purpose search models capable of optimizing neural architectures across diverse domains, without the need for domain-specific tuning?*

To answer this question, we introduce LM-Searcher, a novel framework that leverages LLM's reasoning and optimization capabilities for exploring neural architecture, which transforms neural architectures into task-agnostic universal representations, enabling cross-domain search using either off-the-shelf LLMs or a fine-tuned, search-specific LLM. Specifically, architectures are first represented as numerical strings termed **NCode** for simplicity, and inputed into the LLM for architecture encoding. As illustrated in Fig 1, architecture configurations are represented by their sub-module indices in the search space. This encoding method unifies tasks across different domains by representing them as combinatorial optimization problems.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 9408–9421
November 4-9, 2025

For example, in NAS-Bench-201 (Dong and Yang, 2020), each architecture is built from "cell" units with five configurable operations, encoded as: ["0" (zeroize), "1" (skip connection), "2" (1×1 conv)...]. Unique architectures are represented by combinations of these operation codes.

Such numerical encoding of architectures not only bridges the gap between different architecture domains, but also directs the model's attention towards analyzing inter-architecture relations based on their performance, minimizing distractions from domain-specific characteristics. Subsequently, we extract architecture-performance metadata from established NAS benchmarks (Ying et al., 2019; Zela et al., 2020), which contains up to 10^{18} possible *NCode-accuracy* pairs (e.g., [NCode: 333123, Accuracy: 91.45%]). A straightforward approach is to directly use the accuracy of known historical *NCode-accuracy* pairs to predict better-performing architectures.

However, due to the vastness of the neural architecture search space and the large-scale possible combinations of historical and predicted architectures, it is challenging for LLMs to generate the optimal architecture directly. To mitigate this challenge, we reformulate the task as a ranking problem rather than a generation problem, i.e., the LLM is trained to always choose the next best architecture from a candidate pool.

To effectively construct instruction-tuning samples that conform to our proposed encoding scheme and enhance the LLM's capabilities to explore neural architectures, we introduce a novel pruning-based data curation strategy. This strategy prunes a fully-connected supernet to obtain various sub-networks, following the approach of (Pham et al., 2018). Prior works on pruning (Liu et al., 2018b) and weight-sharing NAS (Xu et al., 2019) have demonstrated that architectures within a reduced sub-network can converge faster and achieve performance comparable to, or even better than, their larger counterparts. Based on this, we sample 100–200 architectures from the same subspace for both the history (NCode-performance) and candidate (NCode) sets, and select the highest-performing architecture as the ground truth. In total, we obtain a 228k instruction-tuning dataset for LLM training.

Training on this carefully curated dataset allows our LLM to learn robust and transferable architecture-performance patterns. As a result, our LM-Searcher can effectively search for architectures not only in image classification tasks but also in previously unseen search spaces. Through comprehensive evaluation of LM-Searcher across diverse tasks without specific tuning, we demonstrate the effectiveness of the proposed LM-Searcher model in in-domain CNN architecture search for image classification (e.g. MobileNetV2) and out-of-domain model design (e.g., LoRA configuration for visual segmentation and generation tasks), respectively.

Our main contributions can be summarized as follows:

- To the best of our knowledge, we introduce **NCode**, the first cross-domain numerical string representation for neural architectures, enabling architecture optimization across diverse domains.
- We develop LM-Searcher, a task-agnostic architecture search model trained using a novel pruning-based subspace sampling data curation mechanism.
- We demonstrate the effectiveness of LM-Searcher across both in-domain CNN architecture search and out-of-domain model design tasks, highlighting its flexibility and generalizability.

2 Related Work

2.1 LLMs Solving Complex Problems

Large language models (LLMs), pre-trained on massive datasets, have demonstrated remarkable proficiency in few-shot learning scenarios (Brown et al., 2020). The field has witnessed rapid advancements in LLM development, with notable contributions including models like LLaMA (Touvron et al., 2023), PaLM (Chowdhery et al., 2023), and GPT-4 (OpenAI, 2023). Subsequent research has focused on fine-tuning these models, particularly open-source models like LLaMA, on specialized or synthesized datasets. This has led to significant performance improvements in areas such as code generation (Roziere et al., 2023) and mathematical reasoning (Wang et al., 2024; Gou et al.). Our work builds upon this trend by exploring the potential of LLMs for the complex task of neural architecture search (NAS). Our approach leverages a carefully curated dataset derived from existing NAS benchmark. This allows LM-Searcher to achieve strong

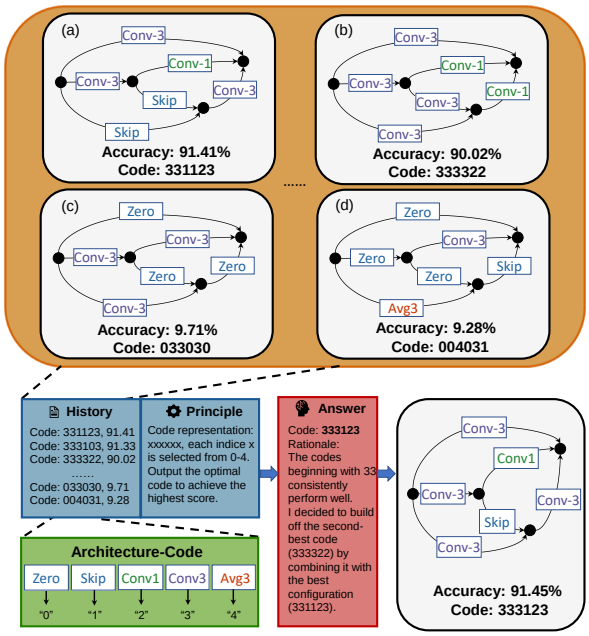


Figure 1: Architecture configurations are encoded as simplified numerical codes and provided to the LLM, which predicts the most promising candidate. This code is then decoded back into a neural architecture for performance evaluation.

cross domain generalization in NAS tasks, setting a new benchmark for LLM performance in this area.

2.2 Architecture Search with LLMs

LLMs have shown promise in supporting model optimization tasks. (Jawahar et al., 2023) utilized GPT-4 (OpenAI, 2023) to estimate the performance of different architectures, using these estimations to guide the initialization of NAS. Similarly, Evo-Prompting (Chen et al., 2024) and LLMatic (Nasir et al., 2024) leverage LLM to perform NAS at the code level. Other studies, such as those by (Song et al., 2024) and (Liu et al., 2024b), have explored the potential of LLMs for black-box optimization in the context of NAS. GENIUS (Zheng et al., 2023) employs GPT-4 to suggest model architectures during the search phase. However, these methods often suffer from limitations such as the need for extensive manual prompt engineering, making them difficult to transfer to different tasks. In contrast, our approach treats the LLM as a general-purpose search model for NAS by introducing a universal representation for encoding architectures across diverse tasks and a pruning-based data construction technique for training the LLM, enabling cross-domain generalization without the need for specific tuning.

3 Method

We introduce LM-Searcher, a task-agnostic neural architecture search framework powered by LLMs. By encoding cross-domain architectures into a unified representation, LM-Searcher leverages LLMs to iteratively analyze historical architecture-performance data, rank new candidates, and predict the most promising architectures until convergence. We begin by formally defining the problem to better leverage LLMs for NAS problems in Sec. 3.1. Then, we present the proposed LM-Searcher pipeline in Sec. 3.2, which consists of three key components: Domain-agnostic Architecture Representation, Optimization Trajectory Data Curation, and Cross-domain Inference.

3.1 Problem Formulation

Neural Architecture Search (NAS) involves finding an optimal neural network architecture from a predefined search space to maximize performance on specific tasks (Zoph and Le, 2016). Formally, given a search space S and history of evaluated architecture, the objective is to identify an architecture a^* that maximizes the performance on the target task:

$$a^* = \underset{a \in S}{\text{argmax}}\ P(a), \tag{1}$$

where $P(a)$ denotes the target performance metric (e.g., accuracy or throughput). Utilizing LLMs with strong in-context learning and reasoning abilities, we aim at building a framework applicable to architecture search across domains, we model all NAS tasks in a consistent form by introducing a universal architecture encoding method and reformulating the architecture generation task as a ranking problem, which identify the optimal architecture from candidate architectures pool C, C sampled from entire search space S. The optimization objective of Eq. 1 can be formulated into

$$a^* = \underset{a \in C}{\text{argmax}}\ P(a). \tag{2}$$

This formulation simplifies the process of obtaining the optimal architecture a^* from a subspace of candidates. In the next section, we introduce the details of training data curation for LLMs training and inference.

3.2 LM-Searcher

Domain-agnostic Architecture Representation. To enable domain-agnostic and LLM-friendly rep-

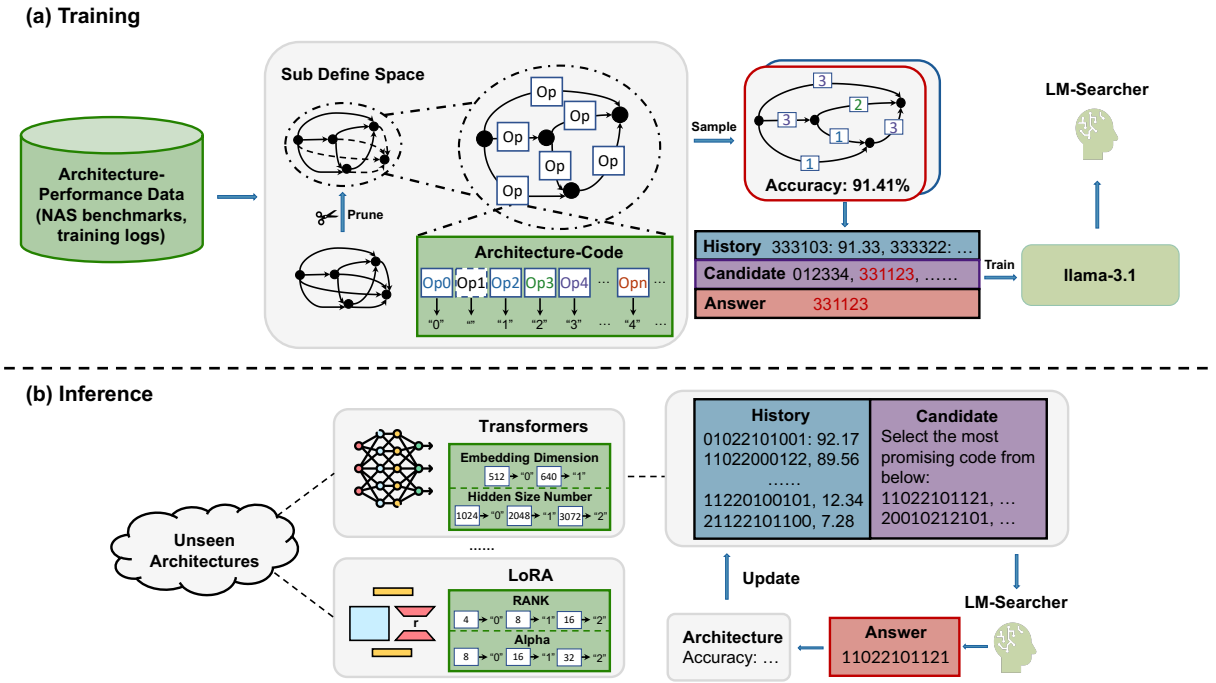


Figure 2: (a) **Training phase:** The full architecture space is pruned into diverse subspaces, from which we sample and encode architectures as numerical codes. Architecture-performance pairs form each training sample, with the top-performing code as the answer. Instructions include a "history" of pairs and a candidate set of codes for the LLM to select from. (b) **Inference phase:** At inference, LM-Searcher frames optimization tasks as combinatorial problems using the same encoding, enabling unified and generalizable search across domains.

resentation of architectures, we encode each architecture as a simplified numerical string called **NCode**. In this encoding, each digit corresponds to the index of a selected configuration option (such as an operator or a topology specification). For instance, in NAS-Bench-201, an architecture is defined by assigning one of five operations—0 (zeroize), 1 (skip connection), 2 (1×1 convolution), 3 (3×3 convolution), or 4 (3×3 average pooling)—to each edge in a directed acyclic graph (DAG) consisting of six edges. Thus, every architecture can be uniquely represented by a six-digit code, where each digit (ranging from 0 to 4) denotes the operation on the corresponding edge (see Fig. 1). Note that some operations, such as zeroize and skip connection, also influence the network topology. For example, an architecture with multiple 3×3 convolutions, a 1×1 convolution, and skip connections would be encoded as "333123", as illustrated in Fig. 1. This approach ensures a compact and cross-domain model representation suitable for LLM processing, enabling the model to analyze relationships between different architectures based on their encoded architecture and corresponding performance.

Optimization Trajectory Data Curation. Leveraging our proposed NCode representation, we can unify and curate large-scale instruction-tuning data from existing NAS benchmarks (Ying et al., 2019; Zela et al., 2020) and training logs, enabling the training of a NAS LLM.

To construct high-quality training samples, we first sparsely prune the entire architecture search space, following (Pham et al., 2018), to generate distinct subspaces for each training subnetwork. As shown in the left part of Fig. 2 (a), each edge in the overall DAG is pruned with a 50% probability, and each selectable configuration is also independently pruned with a 50% probability. This independent pruning process produces diverse subnetworks, thus providing a sufficiently large and varied search space for constructing effective training data for LLMs.

In a pruned subspaces, we randomly sample 100-200 architectures to construct a training sample. However, since the search space is too large to exhaustively explore, retrieving the optimal history-prediction pair is infeasible. To address this, we reformulate the LLM's task: instead of generating the best-performing architecture directly, it ranks a set of candidate architectures sampled from the pruned search space using predefined rules. As

Input
Please analyze the history, rank the candidate and output the highest-performing candidate. **History**: NCode: 03255564, accuracy: 94.28; NCode: 43212502, accuracy: 89.47; ..... NCode: 63421032, accuracy: 25.76; NCode: 53215432, accuracy: 14.13; **Candidate**: 33513501 63225362 ... 41625214
Output
63225362

Table 1: An example of the prompt. The LLM needs to reason and learn from the "history", rank the "candidate", and output which of the "candidate" is expected to perform best.

illustrated in Tab. 1, for each training example, we present the LLM with a history (a list of previous architectures and their accuracies), a set of candidate architectures, and ask it to select the most promising candidate based on patterns learned from the historical data. The training sample thus takes the form: "history (NCode-performance), candidates (NCode), answer (NCode)", where the answer is the actual best-performing architecture among the candidates.

By this means, we generate a dataset comprising 228k optimization trajectory, which are used to fine-tune open-source LLMs and obtain our final LLM-Searcher models.

Cross-domain Inference. During inference, the NCode representation allows us to encode previously unseen architectures into a unified format suitable for processing by the LLM. As illustrated in Fig. 2 (b), for example, when encoding a LoRA module with a search space of [4, 8, 16] for the embedding dimension, the choices 4, 8 and 16 are represented as 0, 1 and 2, respectively.

To demonstrate cross-domain generalization, we evaluate LM-Searcher on both in-domain and out-of-domain tasks *without specifically tuning*. Following the iterative approach used in NAS approaches (Yu et al.; Zoph and Le, 2016), LM-Searcher samples an architecture (NCode) based on historical NCode-performance pairs and 10 randomly generated NCodes. The sampled architecture is then evaluated, and the historical NCode-performance pairs are updated for subsequent iterations.

4 Experiment

4.1 LM-Searcher Training Details

The supervised fine-tuning of LLaMA-3.1-8B on our curated dataset was performed with 320GB of total GPU memory (8×40GB) for 1 epoch. We utilize the codebase provided by (Zheng et al., 2024)[1] to fine-tune the LLaMA-3.1-8B model. The fine-tuning process employs an initial learning rate of 1×10^{-5}, a warm-up step ratio of 0.1, and a batch size of 8 with gradient accumulation steps set to 2. To ensure computational efficiency, we utilize mixed-precision training techniques. Additionally, we leverage DeepSpeed ZeRO-2 (Rajbhandari et al., 2020)[2] to optimize memory usage and enhance scalability during the large-scale training process.

4.2 LM-Searcher for Diverse Tasks

We apply LM-Searcher to perform neural architecture search across diverse domains. For **in-domain** image classification tasks, we evaluate on CIFAR-10, CIFAR-100, and ImageNet-1K (Deng et al., 2009)[3]. Please refer to Appendix A for more details. For **out-of-domain** tasks, in the *image segmentation* task, we use the Kvasir-SEG (Jha et al., 2020) and ISIC2017 (Codella et al., 2018) datasets. For *image generation*, we explore parameter-efficient fine-tuning (PEFT) of diffusion models (Rombach et al., 2022). Additionally, we explore efficient transformer architectures for *machine translation* (Wang et al., 2020) and *audio recognition* using NAS-Bench-ASR (Mehrotra et al., 2021).

The main results are shown in Tab. 2. LM-Searcher exhibits strong search capabilities across diverse domains, including in-domain tasks like CIFAR10 image classification and out-of-domain tasks such as audio recognition, searching efficient transformer architecture for machine translation, and LoRA-based image generation. Compared to other NAS methods, including LLM-based methods, our LM-searcher can not only achieve comparable performance in-domain tasks, but also directly applied to out-of-domain tasks. Additional implementation details and experimental results can be found in Sec. 4.3.

We re-implemented the random search and regularized evolution algorithms to compare their per-

[1] https://github.com/hiyouga/LLaMA-Factory
[2] https://github.com/microsoft/deepspeed
[3] https://www.image-net.org/

Method	Classification (In-domain)			Segmentation		Generation			Translation	Audio
	CIFAR10 Test Err.(%)↓	CIFAR100 Top-1 Acc.(%)↑	ImageNet Top-1 Acc.(%)↑	Kvasir S_α(%)↑	ISIC 2017 Jac(%)↑	Multi-Concept Customization CLIP-T↑	CLIP-I↑	DINO↑	IWSLT'14 De-En BLEU↑	NB-ASR PER(%)↓
DARTS (Liu et al., 2018a)	2.76	15.03	73.3	-	-	-	-	-	-	-
PC-DARTS (Xu et al., 2019)	**2.57**	-	74.9	-	-	-	-	-	-	-
TE-NAS (Chen et al., 2021)	2.63	71.24	73.8	-	-	-	-	-	-	-
AG-Net (Lukasik et al., 2022)	5.63	**73.51**	**76.5**	-	-	-	-	-	-	-
DiNAS (Asthana et al., 2024)	5.63	**73.51**	75.2	-	-	-	-	-	-	-
LLMatic (Nasir et al., 2024)	5.74	71.62	-	-	-	-	-	-	-	-
GENIUS (Zheng et al., 2023)	6.21	70.91	74.9	-	-	-	-	-	-	-
Random Search (Li and Talwalkar, 2020)	3.41	72.07	72.56	91.71	76.28	0.654	0.728	0.385	33.00	21.50
Regularized Evolution (Real et al., 2019)	3.34	71.81	74.5	91.89	77.18	0.659	0.727	0.396	33.45	**21.40**
LM-Searcher	**3.10**	**72.96**	**75.5**	**92.35**	**77.60**	**0.668**	**0.737**	**0.416**	**34.02**	21.44

Table 2: Results of LM-Searcher, compared to various traditional and LLM-based methods on in-domain image classification tasks and 4 out-of-domain tasks (*Image Segmentation, Image Generation, Machine translation, Audio Recognition)*.

formance with our method on out-of-domain tasks. LM-Searcher consistently outperforms evolutionary algorithms (EA) and random search in most scenarios. For in-domain tasks, it achieves performance comparable to specialized NAS methods, while maintaining broader applicability.

4.3 Out-of-Domain Tasks Experimental Details

Image Segmentation. Segment Anything (SAM) (Kirillov et al., 2023) is a foundation model for image segmentation. We employ LM-Searcher to explore the parameter-efficient-finetuning (PEFT) architecture of SAM for image segmentation tasks. We utilize an architecture search space following Conv-LoRA (Zhong et al.)[4], which integrates convolution operations and LoRA modules into SAM's ViT encoder (Dosovitskiy et al., 2020). We employ LM-Searcher to determine effective rank values of 32 LoRA modules from [3, 6, 12, 24]. As rank value of each LoRA module can be configured independently, the total number of unique architectures reaches up to 10^{19}.

During the search phase, we use a proxy task to train and evaluate sampled models on the Kvasir-SEG polyp dataset (Jha et al., 2020) and the skin lesion segmentation dataset (Codella et al., 2019). To ensure time efficiency, training is early stopped after one epoch. The performance of each architecture is assessed using a weighted sum of validation metrics. We conduct the search over 200 iterations and retrain the top-performing model for 30 epochs to evaluate its final performance.

To comprehensively evaluate the effectiveness of the searched architecture across diverse domains, we conduct extensive experiments on multiple downstream tasks. In the medical imaging domain, we validate the architecture's performance by fine-tuning the SAM model on three benchmark datasets: the Kvasir-SEG polyp dataset (Jha et al., 2020) for gastrointestinal polyp segmentation, CVC-612 (Codella et al., 2019) for colorectal lesion analysis, and ISIC2017 (Codella et al., 2018) for skin lesion classification. These experiments systematically demonstrate the architecture's capability in medical domain adaptation and fine-tuning scenarios. For natural image analysis, we assess the model's performance using CAMO (Le et al., 2019) and SBU (Vicente et al., 2016). Furthermore, we extend our evaluation to additional application domains, including agricultural analysis through leaf disease segmentation and remote sensing through road segmentation, thereby demonstrating the searched architecture's versatility across different fields.

Tab. 3 compares the LoRA architecture searched using LM-Searcher with other PEFT models on several segmentation tasks. To account for training variability due to random initialization, each sampled architecture is trained five times, and the average performance is reported. Our LM-Searcher consistently outperforms the Conv-LoRA baseline, achieving superior performance across most evaluation metrics compared to other PEFT approaches. These results indicate that LM-Searcher exhibits strong generalization capabilities across datasets from different domains.

Image Generation. We employs LM-Searcher to identify optimal LoRA configurations for integrating new concepts into a pre-trained Stable Diffusion model (Rombach et al., 2022) following the Mix-of-Show (Gu et al., 2023)[5] setting.We partition the LoRA modules in the Diffusion UNet into 48 distinct groups and explore adaptation ranks

[4] https://github.com/autogluon/autogluon/tree/master/examples/automm/Conv-LoRA

[5] https://github.com/TencentARC/Mix-of-Show

Method	Kvasir		CVC-612		ISIC 2017	CAMO			SBU	Leaf	Road
	$S_\alpha \uparrow$	$E_\phi \uparrow$	$S_\alpha \uparrow$	$E_\phi \uparrow$	Jac↑	$S_\alpha \uparrow$	$E_\phi \uparrow$	$F_\beta^w \uparrow$	BER↓	IoU↑	IoU↑
BitFit (Zaken et al., 2021)	90.8 ± 0.57	93.8 ± 0.98	89.0 ± 0.40	91.6 ± 0.98	76.4 ± 0.45	86.8 ± 0.33	90.7 ± 0.28	81.5 ± 0.19	3.2 ± 0.13	71.4 ± 1.15	60.6 ± 0.15
Adapter (Houlsby et al., 2019)	91.2 ± 0.23	94.0 ± 0.16	89.3 ± 0.43	92.0 ± 0.63	76.7 ± 0.66	87.7 ± 0.10	91.3 ± 0.40	82.8 ± 0.35	2.8 ± 0.09	72.1 ± 0.47	61.5 ± 0.11
VPT (Liu et al., 2024a)	91.5 ± 0.23	94.3 ± 0.06	91.0 ± 0.94	$\mathbf{93.7 \pm 1.41}$	76.9 ± 0.94	87.4 ± 0.60	91.4 ± 0.68	82.1 ± 0.75	2.7 ± 0.06	73.6 ± 0.26	60.2 ± 1.87
LST (Sung et al., 2022)	89.7 ± 0.25	93.3 ± 0.37	89.4 ± 0.37	92.4 ± 0.54	76.4 ± 1.05	83.3 ± 0.28	88.0 ± 0.23	77.1 ± 0.02	3.2 ± 0.01	70.2 ± 0.87	60.2 ± 0.26
SAM-Adapter (Chen et al., 2023)	89.6 ± 0.24	92.5 ± 0.10	89.6 ± 0.22	92.4 ± 1.06	76.1 ± 0.45	85.6 ± 0.26	89.6 ± 0.55	79.8 ± 0.89	3.1 ± 0.06	71.4 ± 0.20	60.6 ± 0.06
SSF (Lian et al., 2022)	91.3 ± 0.87	93.9 ± 1.49	89.6 ± 0.37	91.9 ± 0.79	76.6 ± 0.19	87.5 ± 0.11	91.4 ± 0.16	82.6 ± 0.12	3.2 ± 0.05	71.5 ± 0.63	61.6 ± 0.03
LoRA (Hu et al., 2021)	91.2 ± 0.28	93.8 ± 0.22	90.7 ± 0.04	92.5 ± 0.41	76.6 ± 0.23	88.0 ± 0.24	91.9 ± 0.42	82.8 ± 0.16	2.7 ± 0.08	73.7 ± 0.20	62.2 ± 0.21
Conv-LoRA (Zhong et al.)	91.9 ± 0.64	94.3 ± 0.77	90.1 ± 0.31	92.3 ± 0.42	77.0 ± 0.51	88.6 ± 0.12	92.4 ± 0.17	83.1 ± 0.19	2.8 ± 0.02	73.9 ± 0.20	62.1 ± 0.23
LM-Searcher	$\mathbf{92.4 \pm 0.43}$	$\mathbf{94.7 \pm 0.47}$	$\mathbf{91.1 \pm 0.47}$	93.2 ± 0.70	$\mathbf{77.6 \pm 0.40}$	$\mathbf{88.9 \pm 0.04}$	$\mathbf{92.7 \pm 0.15}$	$\mathbf{83.9 \pm 0.32}$	$\mathbf{2.6 \pm 0.04}$	$\mathbf{74.2 \pm 0.28}$	$\mathbf{62.6 \pm 0.04}$

Table 3: Performance comparison of various PEFT methods across multiple datasets for segmentation tasks. Results are reported as average values with standard errors, calculated over five experimental runs. Notably, LM-Searcher performs NAS in a zero-shot manner without finetuning on the target domain.

Method	CLIP-T	CLIP-I	DINO
P+ (Voynov et al., 2023)	0.686	0.670	0.372
Custom Diffusion (Kumari et al., 2023)	0.650	0.694	0.379
LoRA (Hu et al., 2021)	**0.700**	0.555	0.359
ED-LoRA (Gu et al., 2023)	0.662	0.731	0.404
LM-Searcher	0.668	**0.737**	**0.416**

Table 4: Performance Comparison of PEFT Methods in Multi-Concept Customization for Stable Diffusion Models. Evaluating CLIP-T (Text), CLIP-I (Image), and DINO Metrics. LM-Searcher conducts LoRA configurations search without additional training for image generation tasks.

for each group from the set [4, 8, 16]. In the search phase, LM-Searcher is utilized to sample a model which is subsequently finetuned on 15 images related with a concept. Upon completion of 1,000 training iterations, the model's performance is evaluated using 11 validation prompts, with each prompt generating 8 images for assessment. To quantify the model's capabilities, we utilize the CLIP-Score (Hessel et al., 2021) and the CLIP image alignment score as proxy metrics. These metrics effectively measure the model's caption-following capability and its ability to preserve identity, respectively. The overall performance of an architecture is computed as a weighted sum of two metrics. Upon concluding the search stage, we select the architecture with the highest performance score and proceed to fine-tune plug-and-play LoRA models individually across several concepts. The LoRA models are fused into one SD-V1.5 model with Gradient Fusion (Gu et al., 2023) technique. After fusion, the Stable Diffusion model is tested using 20 prompts per concept, with each prompt generating 50 images.

We report the CLIP text similarity, CLIP image similarity and DINO image similarity of our approach and other PEFT methods in Tab. 4. The model architecture identified by LM-Searcher exhibits superior performance compared to state-of-the-art ED-LoRA across both text similarity and image similarity metrics. These findings indicate that LM-Searcher effectively discovers optimal architectures that not only preserve custom concept identity but also maintain strong image-text alignment. While our model achieves a slightly lower CLIP text score than both P+ (Voynov et al., 2023) and LoRA, it significantly surpasses these approaches in image alignment capability, outperforming P+ by 6.7% and LoRA by 28.2% in CLIP-I.

Efficient Transformer for Machine Translation. To further demonstrate the cross-domain generalizability of LM-Searcher, we apply it to explore efficient transformer architectures for machine translation on edge devices following the HAT (Wang et al., 2020) setting, with a search space encompassing embedding dimensions from [512, 640], hidden dimensions from [1024, 2048, 3072], attention head numbers from [4, 8], and decoder layer numbers from [1, 2, 3, 4, 5, 6]. After training a super-transformer, we sample and evaluate 3K sub-transformers based on their validation losses on the IWSLT'14 De-En validation set under a GPU latency constraint of 500 ms, selecting the sub-transformer with the lowest validation loss for retraining. The retrained model is then evaluated using BLEU and SacreBLEU scores (Post, 2018).

Audio Recognition. To evaluate LM-Searcher's effectiveness in the audio domain. We utilize the tabular NAS-Bench-ASR (Mehrotra et al., 2021) to explore architectures for audio recognition. The search space is modeled by a DAG using four nodes, with each node's configuration consists of a primary operation (e.g., linear operations, convolution operations with various kernel sizes and dilation rates, or a zero operation) and a skip connection operation (e.g., identity or zeroize).

Model	Base Model	NAS-Bench-101	CIFAR100	ImageNet
LM-Searcher-1B	LLaMA-3.1-1B	93.85	72.31	73.52
LM-Searcher-3B	LLaMA-3.1-3B	93.85	72.60	73.93
LM-Searcher-8B	LLaMA-3.1-8B	**93.89**	**72.96**	**75.5**

Table 5: Performance metrics for LM-Searcher with different base model size.

Mapping	CIFAR10		CIFAR100		ImageNet16-120	
	Validation	Test	Validation	Test	Validation	Test
Shuffle	91.22	93.87	72.20	71.89	45.93	46.25
w/o Shuffle	**91.52**	**94.20**	**72.82**	**72.96**	**46.48**	**46.51**

Table 6: Ablation on whether to shuffle the mapping between architectures and performance.

5 Ablation Studies

In this section, we conduct ablation studies to analyze the impact of key components in our LM-Searcher framework. Specifically, we evaluate how our proposed pruning-based subspace sampling and task reformulation affect the overall performance in Sec. 5.1. We also study the impact of the LLM sizes in Sec. 5.2 and whether shuffling the mappings between architectures and performances affects the results in Sec. 5.3.

5.1 Effects of Training Strategy

We assess the effects of our proposed training strategy by evaluating LLaMA-3.1-8B, fine-tuned using datasets constructed with different settings. Specifically, we compare our full training strategy (denoted as "Ours") with two ablation variants: (1) w/o pruning-based subspace sampling, where all the samples in the training data are sampled from the entire search space instead of from a pruned sub search space, and (2) w/o task reformulation, where the LLM is prompted to directly generate better architectures instead of ranking from the candidates. As shown in Tab. 7, our task reformulation technique leads to significant performance improvements on the CIFAR10, CIFAR100, and ImageNet16-120 datasets. Additionally, the proposed pruning-based sampling technique further enhances the LLM's ability to search for neural architectures more effectively.

5.2 Impacts of LLM Size

To investigate the impact of LLM size on performance, we evaluate LM-Searcher with different model scales, with results shown in Tab. 5. We train models at different scales and report test accuracy on NAS-Bench-101, CIFAR-100, and ImageNet. Our results show a consistent improvement as the LLM size increases, indicating that larger models can better capture the promising architecture patterns.

Config	CIFAR10		CIFAR100		ImageNet16-120	
	Validation	Test	Validation	Test	Validation	Test
w/o pruning-based sampling	90.83	93.39	71.08	70.98	44.83	43.98
w/o task reformulation	89.10	92.16	69.33	69.51	43.38	42.53
Ours	**91.52**	**94.20**	**72.82**	**72.96**	**46.48**	**46.51**

Table 7: Performance on NAS-Bench-201 using different training strategy, we report the average accuracy of 5 runs.

5.3 Architecture-performance Mapping

To further validate LM-Searcher's capability in identifying efficient architectures through pattern analysis of historical experimental data, we conducted an ablation study where we intentionally randomized the architecture-performance mapping in NAS-Bench-201. This manipulation introduces noise into the information provided to the LLM. As depicted in Tab. 6, the performance degrades significantly when the architecture-performance pairs provided to LM-Searcher are corrupted by noise. This result shows that LM-Searcher is able to utilize Ncode-performance pairs provided in the context to effectively search architectures. More experimental analysis are provied in Appendix C.

5.4 Trade-off Between Specialization and Generalization

We conducted in-domain fine-tuning with LM-Searcher by encoding architectures with more domain-specific information rather than simple digit encoding. For example, a structure originally encoded as `333123` is now encoded as `|conv3X3~0|+|conv3X3~0|+|conv3X3~1|+|skip_connect~0|+|conv1X1~1|+|conv3X3~2|`, where the numbers following the tilde (~) indicate which previous feature (node) the operator (edge in the DAG) connects to. As shown in Tab. 8, after fine-tuning on 1000 samples created using this encoding method, LM-Searcher achieves performance that approaches or even surpasses existing state-of-the-art methods. However, finetuning the model with such domain-specific samples restricts cross-domain generalization, resulting in performance degradation in other domains (0.97% decline on IWSLT'14 De-En machine translation and 0.13% decline on NAS-Bench-ASR).

Method	CIFAR10↑	CIFAR100↑	ImageNet-16↑	IWSLT'14De-En↑	NB-ASR↓
DARTS (Liu et al., 2018a)	54.30	39.77	16.32	-	-
SGNAS (Huang and Chu, 2021)	93.53	70.31	44.98	-	-
DiNAS (Asthana et al., 2024)	**94.37**	**73.51**	45.51	-	-
AG-Net (Lukasik et al., 2022)	**94.37**	**73.51**	46.42	-	-
LM-Searcher-FT	94.36	**73.51**	**46.54**	33.05	21.57
LM-Searcher	94.20	72.96	46.51	**34.02**	**21.44**

Table 8: Performance on in-domain image classification (CIFAR-10, CIFAR-100, ImageNet-16 from NAS-Bench-201) and out-of-domain tasks (IWSLT'14 De–En machine translation, NB-ASR speech recognition). LM-Searcher-FT is fine-tuned on 1,000 domain-specific samples.

6 Conclusion

In this work, we introduced LM-Searcher, a general-purpose framework for neural architecture optimization that leverages LLMs and a novel cross-domain numerical encoding, NCode. By reformulating neural architecture search as a ranking problem and employing a pruning-based subspace sampling strategy for data curation, LM-Searcher demonstrates robust performance across both in-domain and out-of-domain tasks. Our results highlight the potential of LLMs to serve as flexible, domain-agnostic architecture search models, reducing reliance on domain-specific expertise and manual tuning. We believe our approach sheds new insights into scalable and adaptable methods in automated model search and design. For future work, we aim to expand the training data to encompass broader search spaces and enhance LM-Searcher's efficiency during inference.

7 Acknowledgments

This project is funded in part by National Key R&D Program of China Project 2022ZD0161100, by the Centre for Perceptual and Interactive Intelligence (CPII) Ltd under the Innovation and Technology Commission (ITC)'s InnoHK, and in part by NSFC-RGC Project N_CUHK498/24.

8 Limitations

While LM-Searcher demonstrates promising generality and flexibility across a range of neural architecture search domains, several limitations remain. First, our current focus has been on validating the cross-domain capabilities and generalization of the search model, rather than achieving state-of-the-art (SOTA) performance on specific benchmarks. As a result, there is still a performance gap between LM-Searcher and highly specialized NAS methods tailored for individual domains. Additionally, our approach relies on the availability and quality of existing architecture-performance metadata, which may limit its applicability to domains lacking comprehensive benchmarks. Finally, the NCode representation, while effective for unifying diverse tasks, may not capture all the nuanced architectural details needed for highly specialized designs. We plan to address these limitations in future work by optimizing the model for SOTA performance, expanding to new domains, and further refining the architecture encoding scheme.

References

Rohan Asthana, Joschua Conrad, Youssef Dawoud, Maurits Ortmanns, and Vasileios Belagiannis. 2024. Multi-conditioned graph diffusion for neural architecture search. *arXiv preprint arXiv:2403.06020*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Angelica Chen, David Dohan, and David So. 2024. Evoprompting: Language models for code-level neural architecture search. *Advances in Neural Information Processing Systems*, 36.

Tianrun Chen, Lanyun Zhu, Chaotao Deng, Runlong Cao, Yan Wang, Shangzhan Zhang, Zejian Li, Lingyun Sun, Ying Zang, and Papa Mao. 2023. Sam-adapter: Adapting segment anything in underperformed scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3367–3375.

Wuyang Chen, Xinyu Gong, and Zhangyang Wang. 2021. Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. *arXiv preprint arXiv:2102.11535*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and 1 others. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Noel Codella, Veronica Rotemberg, Philipp Tschandl, M Emre Celebi, Stephen Dusza, David Gutman, Brian Helba, Aadi Kalloo, Konstantinos Liopyris, Michael Marchetti, and 1 others. 2019. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic). *arXiv preprint arXiv:1902.03368*.

Noel CF Codella, David Gutman, M Emre Celebi, Brian Helba, Michael A Marchetti, Stephen W Dusza, Aadi Kalloo, Konstantinos Liopyris, Nabin Mishra, Harald

Kittler, and 1 others. 2018. Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (isbi), hosted by the international skin imaging collaboration (isic). In *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, pages 168–172. IEEE.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

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

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, and 1 others. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Minlie Huang, Nan Duan, Weizhu Chen, and 1 others. Tora: A tool-integrated reasoning agent for mathematical problem solving. In *The Twelfth International Conference on Learning Representations*.

Yuchao Gu, Xintao Wang, Jay Zhangjie Wu, Yujun Shi, Yunpeng Chen, Zihan Fan, Wuyou Xiao, Rui Zhao, Shuning Chang, Weijia Wu, and 1 others. 2023. Mix-of-show: decentralized low-rank adaptation for multi-concept customization of diffusion models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 15890–15902.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. 2021. Clipscore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7514–7528.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Sian-Yao Huang and Wei-Ta Chu. 2021. Searching by generating: Flexible and efficient one-shot nas with architecture generator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 983–992.

Ganesh Jawahar, Muhammad Abdul-Mageed, Laks VS Lakshmanan, and Dujian Ding. 2023. Llm performance predictors are good initializers for architecture search. *arXiv preprint arXiv:2310.16712*.

Debesh Jha, Pia H Smedsrud, Michael A Riegler, Pål Halvorsen, Thomas De Lange, Dag Johansen, and Håvard D Johansen. 2020. Kvasir-seg: A segmented polyp dataset. In *MultiMedia modeling: 26th international conference, MMM 2020, Daejeon, South Korea, January 5–8, 2020, proceedings, part II 26*, pages 451–462. Springer.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, and 1 others. 2023. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026.

Nupur Kumari, Bingliang Zhang, Richard Zhang, Eli Shechtman, and Jun-Yan Zhu. 2023. Multi-concept customization of text-to-image diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1931–1941.

Trung-Nghia Le, Tam V Nguyen, Zhongliang Nie, Minh-Triet Tran, and Akihiro Sugimoto. 2019. Anabranch network for camouflaged object segmentation. *Computer vision and image understanding*, 184:45–56.

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

Dongze Lian, Daquan Zhou, Jiashi Feng, and Xinchao Wang. 2022. Scaling & shifting your features: A new baseline for efficient model tuning. *Advances in Neural Information Processing Systems*, 35:109–123.

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

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024a. Visual instruction tuning. *Advances in neural information processing systems*, 36.

Tennison Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. 2024b. Large language models to enhance bayesian optimization. *arXiv preprint arXiv:2402.03921*.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2018b. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*.

Jovita Lukasik, Steffen Jung, and Margret Keuper. 2022. Learning where to look–generative nas is surprisingly efficient. In *European Conference on Computer Vision*, pages 257–273. Springer.

Abhinav Mehrotra, Alberto Gil CP Ramos, Sourav Bhattacharya, Łukasz Dudziak, Ravichander Vipperla, Thomas Chau, Mohamed S Abdelfattah, Samin Ishtiaq, and Nicholas Donald Lane. 2021. Nasbench-asr: Reproducible neural architecture search for speech recognition. In *International Conference on Learning Representations*.

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

OpenAI. 2023. Https://openai.com/index/gpt-4/.

OpenAI. 2024. Https://openai.com/o1/.

William Peebles and Saining Xie. 2023. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205.

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

Matt Post. 2018. A call for clarity in reporting bleu scores. *arXiv preprint arXiv:1804.08771*.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE.

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

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, and 1 others. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Xingyou Song, Yingtao Tian, Robert Tjarko Lange, Chansoo Lee, Yujin Tang, and Yutian Chen. 2024. Position paper: Leveraging foundational models for black-box optimization: Benefits, challenges, and future directions. *arXiv preprint arXiv:2405.03547*.

Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. 2022. Lst: Ladder side-tuning for parameter and memory efficient transfer learning. *Advances in Neural Information Processing Systems*, 35:12991–13005.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Tomás F Yago Vicente, Le Hou, Chen-Ping Yu, Minh Hoai, and Dimitris Samaras. 2016. Large-scale training of shadow detectors with noisily-annotated shadow examples. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI 14*, pages 816–832. Springer.

Andrey Voynov, Qinghao Chu, Daniel Cohen-Or, and Kfir Aberman. 2023. p+: Extended textual conditioning in text-to-image generation. *arXiv preprint arXiv:2303.09522*.

Hanrui Wang, Zhanghao Wu, Zhijian Liu, Han Cai, Ligeng Zhu, Chuang Gan, and Song Han. 2020. Hat: Hardware-aware transformers for efficient natural language processing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7675–7688.

Ke Wang, Houxing Ren, Aojun Zhou, Zimu Lu, Sichun Luo, Weikang Shi, Renrui Zhang, Linqi Song, Mingjie Zhan, and Hongsheng Li. 2024. Mathcoder: Seamless code integration in llms for enhanced mathematical reasoning. In *12th International Conference on Learning Representations (ICLR 2024)*.

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

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2023. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*.

Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. 2019. Nas-bench-101: Towards reproducible neural architecture search. In *International conference on machine learning*, pages 7105–7114. PMLR.

Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *International Conference on Learning Representations*.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*.

Arber Zela, Julien Siems, Lucas Zimmer, Jovita Lukasik, Margret Keuper, and Frank Hutter. 2020. Surrogate nas benchmarks: Going beyond the limited search spaces of tabular nas benchmarks. *arXiv preprint arXiv:2008.09777*.

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

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*.

Zihan Zhong, Zhiqiang Tang, Tong He, Haoyang Fang, and Chun Yuan. Convolution meets lora: Parameter efficient finetuning for segment anything model. In *The Twelfth International Conference on Learning Representations*.

Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.

A In-Domain Tasks Experimental Details

NAS-Bench-201 NAS-Bench-201 [6] (Dong and Yang, 2020) is a widely used benchmark that provides pre-computed results of 15625 architectures pretrained and evaluated on three different image classification datasets.

ImageNet-1K Dataset The ImageNet-1K [7] dataset is a widely used benchmark in computer vision, particularly for image classification tasks. It comprises images from 1,000 distinct object categories. The training set contains approximately 1.2 million labeled images, while the validation set includes 50,000 images, with 50 samples per class.

We evaluate LM-Searcher's effectiveness on searching architectures for image classification under the large-scale DARTS search space. On the CIFAR-10 dataset, we employ an efficient proxy task—training each sampled architecture for only 10 epochs with a cosine learning rate scheduler. During the search phase, LM-Searcher samples 500 unique architectures. We then fully retrain the top 5 models for 200 epochs on CIFAR-10 training set to identify the highest performing architecture. On the ImageNet dataset, we utilize LM-Searcher to sample a total of 1K architectures from the surrogate benchmark (Zela et al., 2020). We retrain the top performing model on ImageNet training set for 250 epochs. Both training on CIFAR-10 and ImageNet use the SGD optimizer with hyperparameters following (Liu et al., 2018a). Test set accuracy is reported and compared with other NAS methods that search architectures on CIFAR-10 and retrain on ImageNet. For a fair comparison, we re-implement GENIUS on CIFAR-10 using the official codebase[8].

To benchmark LM-Searcher's performance against other NAS approaches. We utilize the widely-used NAS-Bench-201 (Dong and Yang, 2020). To mitigate the effects of randomness, each search experiment is conducted over five independent runs. As depicted in Table 9, LM-Searcher achieves competitive performance across all the dataset.

B Metric Explanation

S-measure (S_α) S-measure evaluates the structural similarity between the predicted segmentation and ground truth. It considers both region-aware (S_r) and object-aware (S_o) structural similarities:

$$S_\alpha = \alpha \cdot S_o + (1 - \alpha) \cdot S_r$$

where α is typically set to 0.5. The object-aware similarity focuses on the foreground object consistency, while region-aware similarity evaluates the overall structural information.

E-measure (E_ϕ) The Enhanced-alignment measure captures both pixel-level matching and image-level statistics. It is computed as:

$$E_\phi = \frac{1}{W \times H} \sum_{x=1}^{W} \sum_{y=1}^{H} \phi_{FM}(x, y)$$

where ϕ_{FM} is the enhanced alignment matrix, and W, H are the width and height of the image.

Jaccard Index (Jac) Jaccard Index is also known as Intersection over Union (IoU), it measures the overlap between predicted and ground truth regions:

$$\text{Jaccard} = \frac{|P \cap G|}{|P \cup G|}$$

where P is the predicted segmentation and G is the ground truth.

[6] https://github.com/D-X-Y/NAS-Bench-201

[7] https://www.image-net.org/

[8] https://github.com/mingkai-zheng/GENIUS

F-measure (F_β^ω) The weighted F-measure combines precision and recall with adaptive weights:

$$F_\beta^\omega = \frac{(1+\beta^2)\cdot \text{Precision}^\omega \cdot \text{Recall}^\omega}{\beta^2 \cdot \text{Precision}^\omega + \text{Recall}^\omega}$$

where the superscript ω indicates weighted versions that emphasize errors in important regions.

Boundary Error Rate (BER) The boundary error rate specifically evaluates the accuracy of boundary detection, calculated as:

$$\text{BER} = \frac{|B_P \setminus B_G| + |B_G \setminus B_P|}{|B_G|}$$

where B_P and B_G are the predicted and ground truth boundaries, respectively.

Bilingual Evaluation Understudy (BLEU) Bilingual Evaluation Understudy (BLEU) is an automatic metric for evaluating machine translation and other text generation tasks by measuring how closely a system's output matches one or more reference translations. It computes modified n-gram precision: counts of overlapping 1- to 4-grams between the candidate and references, clipped so repeated n-grams aren't over-rewarded. These precisions are combined via a geometric mean and multiplied by a brevity penalty that downweights outputs shorter than the references.

Phoneme Error Rate (PER). Phoneme Error Rate (PER) is a standard metric for evaluating phoneme-level recognition or alignment systems. Given a reference phoneme sequence and a hypothesized sequence, PER is computed as the edit distance between the two sequences—decomposed into substitutions (S), deletions (D), and insertions (I)—normalized by the length of the reference sequence (N). Formally,

$$\text{PER} = \frac{S + D + I}{N} \times 100\%,$$

where S, D, and I are obtained via an optimal alignment (e.g., dynamic programming for Levenshtein distance).

C Experimental Analysis

C.1 LM-Searcher Search behavior

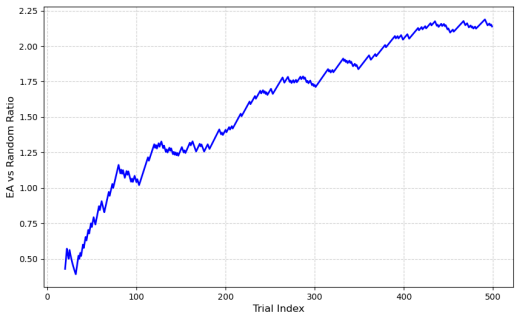
In this section, we investigate the selection behavior of LM-Searcher by analyzing which architectures it tends to prefer. To this end, we generate half of the candidate architectures using a random search algorithm and the other half using evolutionary algorithms (EA). During the search phase, we track whether LM-Searcher selects candidate architectures originating from random search or EA. Specifically, we report the ratio of random search selections to EA selections as a function of the trial iteration number. As depicted in Fig. 3, in the earlier stage of searching, LM-Searcher choose more candidates generated by random search than EA. As the search continues to iterate, the proportion of EA selection grows up with the trial iteration number, and surpassing the portion of random search selection after approximately 50-100 trials. A similar trend is observed when searching on the CIFAR-10, CIFAR-100, and ImageNet-16 datasets.

Method	CIFAR10		CIFAR100		ImageNet16-120	
	Validation	Test	Validation	Test	Validation	Test
DARTS (Liu et al., 2018a)	39.77	54.30	15.03	15.61	18.87	16.32
SGNAS (Huang and Chu, 2021)	90.18	93.53	70.28	70.31	44.65	44.98
AG-Net (Lukasik et al., 2022)	91.60	**94.37**	**73.49**	**73.51**	**46.73**	**46.42**
DiNAS (Asthana et al., 2024)	**91.61**	**94.37**	**73.49**	**73.51**	46.66	45.41
LLM-based						
GENIUS (Zheng et al., 2023)	91.07	93.79	70.96	70.91	45.29	44.96
LLMatic (Nasir et al., 2024)	-	**94.26**	-	71.62	-	45.87
LM-Searcher	**91.52**	94.20	**72.82**	**72.96**	**46.48**	**46.51**

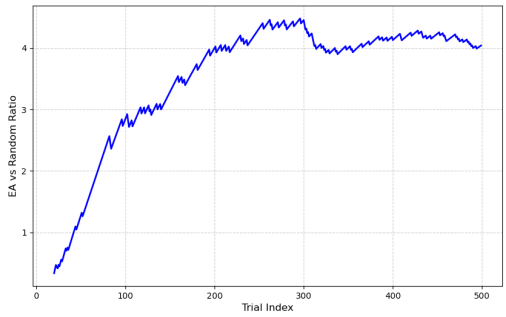
Table 9: Comparison with specialized or LLM-based NAS methods on NAS-Bench-201. The result of Random Search was reported by (Lukasik et al., 2022).

C.2 LM-Searcher Attention Pattern

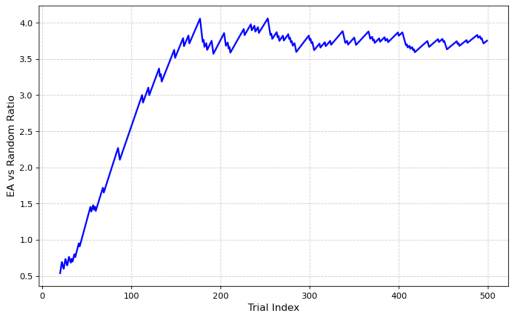
We visualize the attention maps produced by LM-Searcher during the search process on NAS-Bench-201. Table 10 reports the Pearson correlation coefficients between attention scores (measured between candidate NCode tokens and history NCode tokens) and two specific attributes of the history tokens: their performance and their similarity to the candidate NCode. As shown in the table, the attention scores exhibit a higher Pearson correlation with the performance of history NCode tokens than with their string similarity to the candidate NCode. This indicates that LM-Searcher does not solely depend on the external similarity to predict architectures. Notably, the negative correlation between attention scores and performance indicates that architectures with lower performance may provide more valuable information for efficient exploring the architecture search space.



(a) CIFAR10 dataset



(b) CIFAR100 dataset



(c) ImageNet-16 dataset

Figure 3

Method	**CIFAR10**		**CIFAR100**		**ImageNet16-120**	
	Pearson coe	P-value	Pearson coe	P-value	Pearson coe	P-value
NCode similarity	-0.0305	0.2953	-0.0216	0.3152	-0.0434	0.2829
Performance	-0.3591	0.0000	-0.4022	0.0003	-0.3082	0.0001

Table 10: Correlation coefficients between attention score and architecture attributes.