Design Principle Transfer in Neural Architecture Search via Large Language Models

Xun Zhou[1], **Xingyu Wu**[1*], **Liang Feng**[2*], **Zhichao Lu**[3], **Kay Chen Tan**[1]

[1]Department of Data Science and Artificial Intelligence, The Hong Kong Polytechnic University
[2]College of Computer Science, Chongqing University
[3]Department of Computer Science, City University of Hong Kong
{xingy.wu, kctan}@polyu.edu.hk, xunzhou6-c@my.cityu.edu.hk, liangf@cqu.edu.cn, luzhichaocn@gmail.com

Abstract

Transferable neural architecture search (TNAS) has been introduced to design efficient neural architectures for multiple tasks, to enhance the practical applicability of NAS in real-world scenarios. In TNAS, architectural knowledge accumulated in previous search processes is reused to warm up the architecture search for new tasks. However, existing TNAS methods still search in an extensive search space, necessitating the evaluation of numerous architectures. To overcome this challenge, this work proposes a novel transfer paradigm, i.e., design principle transfer. In this work, the linguistic description of various structural components' effects on architectural performance is termed design principles. They are learned from established architectures and then can be reused to reduce the search space by discarding unpromising architectures. Searching in the refined search space can boost both the search performance and efficiency for new NAS tasks. To this end, a large language model (LLM)-assisted design principle transfer (LAPT) framework is devised. In LAPT, LLM is applied to automatically reason the design principles from a set of given architectures, and then a principle adaptation method is applied to refine these principles progressively based on the new search results. Experimental results show that LAPT can beat the state-of-the-art TNAS methods on most tasks and achieve comparable performance on others.

Introduction

Neural architecture search (NAS) has recently become a prominent research focus for the automatic construction of deep neural networks (DNNs) (Zhou et al. 2021b). With the advancement of NAS methods, automatically generated DNNs surpass manually crafted ones across various deep learning tasks (Zhou et al. 2021a). Nonetheless, the majority of NAS methods concentrate on constructing DNN architectures for a single task. This implies that these methods must run repeatedly from scratch when networks for multiple tasks are required. This constraint impedes the practical utility of NAS in real-world settings.

To address this limitation, transferable NAS (TNAS) has been introduced (Zhou et al. 2023; Li et al. 2021; Elsken et al. 2020; Lu et al. 2021; Huang et al. 2022; Liu, Simonyan, and Yang 2018). TNAS leverages architectural knowledge gained from prior search processes to expedite the architecture search for new tasks, effectively "warming up" the search process. For instance, previously discovered architectures can inform and guide the search toward more promising configurations for a new task (Zhou et al. 2023; Li et al. 2021; Elsken et al. 2020). Similarly, performance predictors built during earlier NAS tasks can be reused to reduce the computational cost of evaluating new architectures (Lu et al. 2021; Huang et al. 2022). Nevertheless, these approaches still operate within vast search spaces (e.g., 10^{18} candidate architectures in DARTs (Liu, Simonyan, and Yang 2018)), necessitating the evaluation of a prohibitively large number of architectures and thus resulting in an inefficient search process. Additionally, the current TNAS methods often intertwine the utilization of transferred knowledge with specific search techniques, limiting their adaptability to different NAS methodologies.

This paper introduces a novel paradigm called design principle transfer, aimed at overcoming these challenges. In this approach, design principles—expressed as linguistic descriptions of how certain structural components (e.g., layers or connections) affect the performance of architectures—are first extracted from architectures developed for previous tasks. These principles are then applied to prune the search space by eliminating architectures with less critical components, resulting in an optimized subspace for the new task. By searching within this refined subspace, the proposed approach aims to significantly enhance both the efficiency and performance of NAS for new tasks. Importantly, because knowledge reuse is decoupled from the architecture search itself, this paradigm is compatible with most NAS methods.

Despite its promise, the practical implementation of design principle transfer presents several challenges. Firstly, the complexity and diversity of DNN architectures make it difficult to distill general design principles. Current methods often require specialized tools to map architectures into a shared latent space, followed by expert analysis to extract underlying design rules (Yuan et al. 2022), which reduces the level of automation. Secondly, the process of learning these principles is resource-intensive, requiring a vast number of labeled architectures. For instance, Radosavovic et al. (2020) trained over 500 architectures to investigate the relationship between architectural width and performance,

*Corresponding author

incurring costs that exceed those of most NAS efforts. Additionally, the high-level abstraction of knowledge in natural language complicates its translation into actionable insights for architecture design.

With the emergence of pre-trained Large Language Models (LLMs) (Wu et al. 2024; Liu et al. 2024), LLMs offer a promising solution to address the aforementioned challenges. By representing architectures in programming languages, the task of learning design principles can be framed as a language-inductive learning problem, a domain where LLMs have demonstrated proficiency (Imani, Du, and Shrivastava 2023). Therefore, leveraging LLMs as reasoning agents for automatic design principle learning is a logical step. Given their pre-training on vast knowledge, in-context learning can be employed to tackle this task, thereby mitigating the constraints posed by the number of architecture samples. Furthermore, owing to their contextual awareness, LLMs can automatically translate design principles into actionable architectural knowledge for NAS methods.

Keeping the above in mind, this work proposes an LLM-assisted design principle transfer (LAPT) framework. The LAPT framework employs a pre-trained LLM to learn design principles from well-established neural architectures represented in programming codes. These principles are then translated into constraints that refine the predefined search space, optimizing it for new NAS tasks. However, due to domain shifts, the refined subspace may not always be optimal for every task. To address this, we introduce a principle adaptation method that refines the design principles based on the architectures found for the target task, thereby building the search space to the specific requirements of this task. The main contributions of this paper are summarized as follows:

- To the best of our knowledge, this work is the first research for the design principle transfer. This novel transfer paradigm aims to build a refined search space for new NAS tasks, leading to the improvement of search performance and efficiency.
- An LLM-assisted framework is proposed to implement the design principle transfer across different NAS tasks, which offers at least three advantages: (i) Learning of the general design principles based on LLMs; (ii) Task-specific principle adaptation against domain shit; (iii) Improved interpretability of search space refinement.
- Extensive experiments across various search spaces and tasks demonstrate the effectiveness of LAPT. Even when using standard NAS methods, searching within the refined search space leads to state-of-the-art (SOTA) results, highlighting the potential of design principle transfer as a promising research direction in NAS.

Related Work

This section reviews the most closely related studies to our work, namely TNAS and the utilization of LLMs in NAS.

TNAS

To reduce the computational cost, transfer learning has been incorporated into NAS, giving rise to TNAS (Huang et al. 2022; Lee and Hyung 2021; Lu et al. 2020, 2021; Elsken et al. 2020; Zhou et al. 2023). In TNAS, the knowledge gained during the architecture search for one task is applied to aid in the architecture design for others. Depending on the types of transferred knowledge, TNAS can be classified into two categories. One category transfers already found architectures from previous tasks to directly build architectures for new tasks. For instance, Lu et al. (Lu et al. 2020) design architectures for CIFAR-100 and use them to classify images from ImageNet directly. However, the generalization performance of these methods has been criticized. To solve this drawback, multitasking evolutionary NAS (MT-NAS) (Zhou et al. 2023) applies these high-performing architectures to guide the architecture search for new tasks instead of solving them directly.

The second category reuses the NAS model built in previous tasks to guide the architecture search for a new task. For example, Lu et al. (2021) leverage the supernet that has been initialized on ImageNet to improve the search efficiency of evolutionary NAS (ENAS) on new image classification tasks. In another related work (Elsken et al. 2020), a meta-architecture is acquired from previous tasks, and this architecture can be adapted to a new task just through a few gradient-based steps. Huang et al. (2022) fine-tune the performance predictor developed for the Jigsaw task and use it to reduce the search cost on other computer vision tasks. However, these methods still search in a very large search space, consuming a lot of computing resources for obtaining high-quality architectures. In this work, a new paradigm of TNAS is proposed, i.e., design principle transfer. In this method, general design principles are summarized from established architectures and further reused to reduce the search space for new NAS tasks.

LLMs For NAS

Most existing NAS methods still follow a semi-automated process, where designs for search space, search method, and performance evaluation system rely on expert knowledge (Chen, Dohan, and So 2024). To improve the automation degree of architecture design, LLMs have been incorporated into NAS, because of their robust capabilities in solving various domain-specific tasks. For instance, leveraging its powerful generative capabilities, GPT-4 has been utilized to generate convolution neural networks (CNNs) (Chen, Dohan, and So 2024; Zheng et al. 2023; Nasir et al. 2023; Qin et al. 2024) and graph neural networks (GNNs) (Wang et al. 2023; Dong et al. 2023). When compared to many mainstream NAS methods, these LLM-based methods can achieve comparable performance. Moreover, LLMs can serve as a performance predictor within NAS to accelerate the search process (Zhang et al. 2023; Jawahar et al. 2023; Chen et al. 2024). In contrast to performance predictors developed using machine learning methods or DNNs, LLMs can achieve comparable performance with fewer training samples. Contrary to existing works focusing on the refinement of the search method and the performance evaluation system, our work applies LLMs to refine search space, further extending the application of LLMs in NAS research.

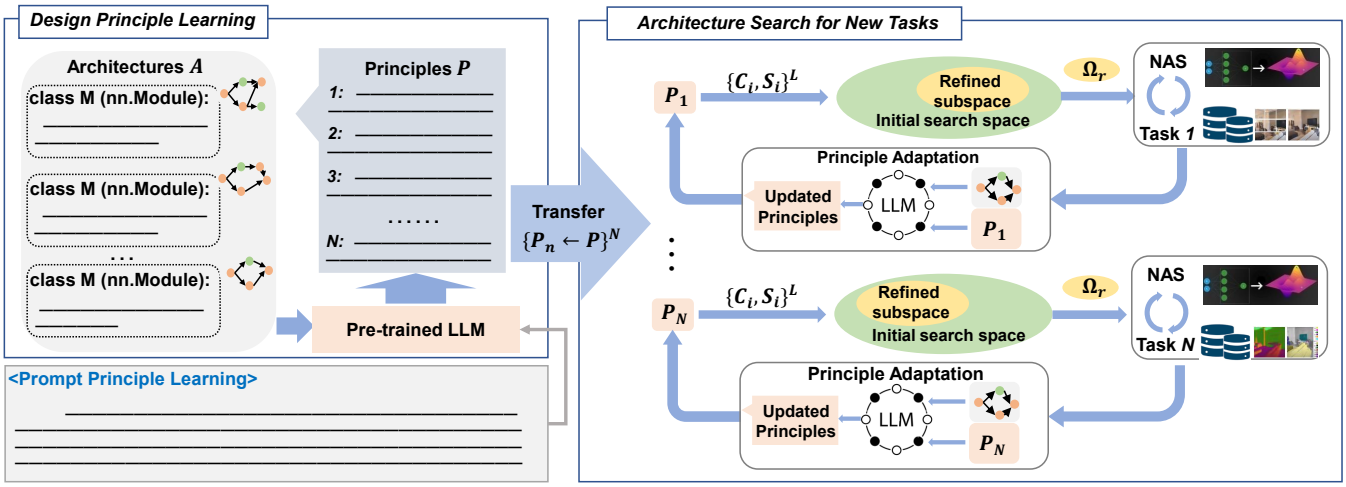

Figure 1: Overview of the proposed LAPT. This framework consists of two stages. In the learning stage of design principles, LLM is driven by specific prompts to learn general design principles from a set of architectures. In the architecture search stage, the learned principles are applied to initialize the search space for each new task. Then, architectures found in the refined search space are used to update these principles, aiming to build the task-specific search space.

Proposed Method

In this section, we initially define the architecture search problem and introduce a design principle transfer framework to effectively address it. Subsequently, details of these key components within this framework are presented including design principle learning and principle adaptation.

Problem formulation

Given a task $\mathcal{T}$ associated with labeled dataset $\mathcal{D}$, we consider searching an architecture a^* from a predefined search space Ω to achieve the best accuracy (termed $ACC.$) on this task. Different from existing NAS works that directly search in the entire search space, we only explore a promising subspace Ω_r for the efficient architecture search. This NAS problem can be formulated as follows:

$$\begin{aligned} a^* = &\arg\max_{a \in \Omega_r} ACC.(a,\ \mathcal{D}) \\ &s.t.\ \Omega_r \subset \Omega, \end{aligned} \tag{1}$$

where the key to solving equation (1) is building the optimal subspace Ω_r of Ω. In this work, Ω consists of a set of available architectures for the target task $\mathcal{T}$. For each architecture A in Ω, it follows a feed-forward structure with no more than L layers. Each layer l_i is associated with some operators (such as pooling and convolution) to deal with the received feature information, where $\mathcal{C}_i$ denotes the candidate operators for the ith layer. Additionally, the feed-forward structure makes l_i only receive information from its previous layers $\mathcal{S}_i = \{l_1, \ldots, l_{i-1}\}$, where s_i denotes the information sources of the ith layer. Thus, an architecture A can be parameterized as $A = \cup_{i=1}^{L}\{l_i,\ s_i\}$, and Ω can be formulated as equation (2):

$$\Omega = \{A |\ l_i \in \mathcal{C}_i,\ s_i \subset \mathcal{S}_i, i \in \{1, \ldots,\ L\}\}. \tag{2}$$

To extract the optimized subspace from Ω, design principles that describe the influence of various operators and connections for the architecture performance can be used to discard the unimportant operators and information sources of each layer. In this way, a subspace for each layer, i.e., refined candidate operators $\mathcal{C}_i^r \subset \mathcal{C}_i$ and refined candidate information sources $\mathcal{S}_i^r \subset \mathcal{S}_i$, can be built. The refined search space Ω_r is in equation (3):

$$\Omega_r = \{a |\ l_i \in \mathcal{C}_i^r,\ \ s_i \subset \mathcal{S}_i^r,\ a \in \Omega\}. \tag{3}$$

Compared with Ω, Ω_r has a higher proportion of well-performing architectures, searching in this optimized search space is more efficient to reach the architecture with good performance. The quality analysis of Ω and Ω_r is presented in https://arxiv.org/abs/2408.11330.

Framework

An overview of LAPT is presented in Figure 1 and Algorithm 1. This framework consists of two stages, i.e., design principle learning and the architecture search for new tasks. In the design principle learning stage, a set of well-performing architectures $\mathcal{A}$ is collected from a pre-defined search space Ω (line 1 in Algorithm 1). Then, a prompt is devised to help a pre-trained LLM reason general design principles P from $\mathcal{A}$ (line 2 in Algorithm 1). These learned principles P are transferred as the initial design principles $\{P_n\}_{n=1}^N$ to help solve N new NAS tasks $\{\mathcal{T}_n\}_{n=1}^N$ (lines 3-10 in Algorithm 1). Specifically, for the task $\mathcal{T}_n$, P_n is translated to the subsets $\{\mathcal{C}_i^r,\ \mathcal{S}_i^r\}_i^L$ by the pre-trained LLM, leading to the generation of a refined search space Ω_r. Then, a NAS method is applied to search for the promising architectures $\mathcal{B}_n$ from Ω_r for $\mathcal{T}_n$. These found architectures are applied to adapt P_n to $\mathcal{T}_n$. These steps related to principle adaptation and architecture search are repeated until reaching a predefined stopping criterion. Finally, the best architecture found in the search process is used to solve $\mathcal{T}_n$.

Algorithm 1: Framework of the proposed LAPT

Input: Search space Ω, new tasks $\{\mathcal{T}_n\}_{n=1}^N$, a pre-trained LLM, a NAS method, iterations G.
Output: Architecture $\{a_n^*\}_{n=1}^N$.

Build an archive $\mathcal{A}$ consisting of performing architectures from $Omega$;
$\{P_n\}_{n=1}^N \leftarrow$ Prompt the LLM to learn design principles P from $\mathcal{A}$ and set it as initial principles for each task;
for n from 1 to N **do**
  $Base \leftarrow 0$;
  for g from 1 to G **do**
    $\Omega_r \leftarrow$ LLM translate P_n to a set of subset $\{\mathcal{C}_i^r,\ \mathcal{S}_i^r\}_i^L$ of Ω and build subspace follow (3);
    $\mathcal{B}_n \leftarrow$ Search for architectures for the task $\mathcal{T}_n$ from Ω_r by the given NAS method;
    $Best \leftarrow$ Evaluate architectures $\mathcal{B}_n$ on $\mathcal{T}_n$ and record the performance of the best one;
    $P_n,\ Base \leftarrow$ Adapt P_n based on $\mathcal{B}_n$, $Base$, $Best$ and return the updated principles; #Alg. 2
  end for
end for
Return the best architecture $\{a_n^*\}_{n=1}^N$ found in the search process.

Design principle learning

In this part, a prompt is designed to guide the LLM to learn the design principles from the given architectures. The prompt consists of two parts, i.e., architecture implementation and learning guidelines.

Firstly, the pre-trained LLM benefits from exposure to a wide array of programming languages, allowing it to gain awareness of the neural architecture from source codes (Zheng et al. 2023). Nevertheless, due to the token limitation, it becomes infeasible to feed all architecture source codes directly into the LLM. To tackle this issue, Python classes that can instantiate an architecture from its architectural parameters, i.e., $\cup_{i=1}^L\{l_i,\ s_i\}$, are set as prompts. This approach enables LLMs to assimilate knowledge about these neural architectures solely through a few architecture parameters. A simple example of the Python code-based prompt is shown in Figure. 2.

Secondly, instructing LLMs to reason the general design principles from such architectures is not trivial, given the complex and diverse DNN architectures. To address this issue, drawing inspiration from the effective utilization of the "chain of thought" method in LLMs, we steer the LLM towards a step-by-step reasoning process as follows:

- **Step1:** input architectural parameters of the given architectures into the LLM;
- **Step2:** prompt LLM identifying common patterns within these architectures;
- **Step3:** summarize the design principle behind these common patterns.

More Details of this prompt can be found in https://arxiv.org/abs/2408.11330.

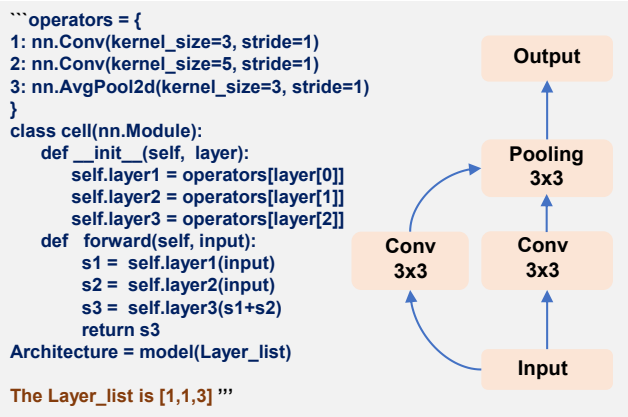

Figure 2: The Python code-based prompt can help LLMs gain awareness of the right CNN architecture from its architectural parameters which are represented as "Layer_list".

Algorithm 2: Principle Adaption

Input: design principles P_n, found architectures $\mathcal{B}_n$, $Base$, $Best$, # of selection architectures r.
Output: updated P_n.

if $Base \leq Best$ **then**
  $P_n \leftarrow$ Prompt LLM to update P_n from the top r architectures of $\mathcal{B}_n$;
  $Base \leftarrow Best$;
else
  $P_n \leftarrow$ Prompt LLM to describe effects of other operators and connections that are not in current P_n;
end if
Return P_n, $Base$.

Principle transfer and adaptation

In the architecture search stage, principles P learned in the last stage are used to help build the refined search space Ω_r. Specifically, we prompt the pre-trained LLM to translate P into the most available candidate operators $\mathcal{C}_i$ and information sources $\mathcal{S}_i$ for each layer. In this way, Ω_r can be built following equation (3). Then, the NAS method is used to search architectures $\mathcal{B}_n$ from Ω_r, aiming to solve the problem shown in equation (1).

However, due to domain shift where the architecture performs differently on different tasks, Ω_r may not be optimal for all the tasks. To alleviate the potential negative effects of domain shift, this work adapts P to the target task based on the newly found architectures. As a result, a task-specific subspace can be built progressively. Specifically, the NAS method is applied to search for architecture from Ω_r in an iterative way. In each iteration, if the better-performing architectures are found, LLM is prompted to update P_n based on these architectures; otherwise, LLM is required to describe effects of other available candidate operators and information sources that are not in P_n, promoting the exploration for other promising regions in Ω. Details of this adaption strategy can be seen in Algorithm 2.

Stage	Parameter	NAS201	Trans101	DARTs
Learning	# of samples	50	50	100
Adaption	r	5	15	50
	# of iterations	3	4	2
REA	population size	5	10	20
	# of generations	1	1	10
	tournament size	2	5	2
	crossover probability	-	-	0.5
	mutation probability	1	1	0.5
Supernet	# of epochs	-	-	30

Table 1: Hyper-parameters settings

Experiments

To demonstrate the efficacy of the proposed LAPT, we perform architecture searches for diverse tasks utilizing the design principles learned from established architectures. Subsequently, we compare its search performance with other well-known NAS and TNAS methods. This section commences with a description of the search spaces and experimental setup, followed by the presentation of comparisons.

Search space

This work tests the proposed TNAS method on three search spaces, i.e., NAS-bench-201 (NAS201) (Dong and Yang 2020), TransNAS-Bench-101 (Trans101) (Duan et al. 2021), and DARTs (Liu, Simonyan, and Yang 2018).

NAS201. The architecture in NAS201 consists of the repeated cells. Each cell consists of 6 layers and there are five candidate operators for each layer. These layers are connected following a predefined pattern leading to 15K architectures in this search space.

Trans101. Architectures in Trans101 follow the same cell-based structure in NAS201, but there are only 4 candidate operators for each layer. Thus, only 4K candidate architectures are included in the space.

DARTs. Architectures in this search space also follow the cell-based structure. Particularly, each architecture consists of stacking two types of cells, i.e., the normal cell and the reduction cell. For each cell, there are 2 input layers and 4 ordered stages. Each stage includes two layers, and each layer has 8 candidate operators to deal with information from previous stages or input layers, resulting in $\prod_{k=1}^{4} \frac{k(k+1)}{2} 8^2 \approx 10^9$ candidate cell structures. Since we jointly learn both normal and reduction cells, the total number of architectures is $(10^9)^2$.

Experiment Setup

LAPT is employed to search for task-specific architectures within the three distinct search spaces mentioned above. Given that this framework is adaptable to various NAS methods, the vanilla NAS method **Regular EA (REA)** (Real et al. 2019) is utilized in these experiments to showcase the efficacy of the design principle transfer. Details of hyperparameters are in Table 1 and implementations are as follows:

Experiments on NAS201. In these experiments, we collect 50 top-performing architectures on ImageNet from NAS201 as the input for LLM. Then, learned principles are used to constrain the range of available operators on each layer, leading to a refined search space of NAS201. We separately search for top-performing architectures (i.e., Top 0.1%) on CIFAR-10 and CIFAR-100 from the refined search space and record the time cost.

Experiments on Trans101. In these experiments, we collect 50 top-performing architectures on Jigsaw from Trans101 as the input for LLM. Then, learned principles are used to constrain the range of available operators on each layer, leading to a refined search space of Trans101. We search for architectures from the refined search space to solve 6 different computer vision tasks i.e., Object Classification (Obj), Scene Classification (SC), Room Layout (Roo), AutoEncoder (Auto), Surface Normal (Nor), Semantic Segmentation (Seg), respectively.

Experiments on DARTs. In these experiments, we randomly select 5000 architectures from DARTs and evaluate them on CIFAR-100. To reduce the evaluation cost, a supernet that covers all the architectures in DARTs is pre-trained on CIFAR-100 and each architecture can inherit weights from it for evaluation. The top 100 architectures in the selected architectures are the input of LLM. Then, learned principles are used to constrain the number of candidate operators on each layer and its candidate information sources, leading to a refined search space of DARTs. We search for individual architecture from the refined search space to solve image classification tasks on CIFAR-10 and ImageNet. To accelerate the search process, a supernet that covers the refined search space is built and then pre-trained on the target task. As a result, each architecture can inherit weights from it for evaluation.

In these experiments, GPT-4 is used as the pre-trained LLM for design principle learning and adaptation. Related prompts and the effectiveness of different LLMs are shown in https://arxiv.org/abs/2408.11330.

Method	CIFAR-10		CIFAR-100	
	Acc.(%)	**Archs.**	**Acc.**	**Archs.**
REA (Real et al. 2019)	93.92	>500	71.84	>500
HEBO (Cowen-Rivers et al. 2022)	94.34	100	72.62	100
MetaD2A (Lee and Hyung 2021)	94.37	100	73.34	100
TNAS-BO (Shala et al. 2023)	94.37	29	73.51	59
LAPT-REA	**94.36**	**4.9**	**73.45**	**8.6**

Table 2: Comparison with TNAS and NAS methods on NAS201.We present the test accuracy achieved by the found architectures on two unseen datasets. Additionally, we provide the number of neural architectures (**Archs.**) that are trained in the search. We run LAPT 20 times with different random seeds and average values are reported.

Comparison to NAS methods

Comparison on NAS201. We choose well-known NAS methods and TNAS methods as baselines: (1) the vanilla NAS method REA and the state-of-the-art NAS method HEBO (Cowen-Rivers et al. 2022); (2) the state-of-the-art TNAS methods include TNAS-BO (Shala et al. 2023) and

Tasks	Obj	SC	Roo	Auto	Nor	Seg	Total
Metric	Acc↑	Acc↑	L2 loss↓	SSIM↑	SSIM↑	L2 MIoU↑	Ave. Rank↓
REA (Real et al. 2019)	45.39%	54.62%	61.75	**56.96**	57.22	25.52	38.50
BONAS (Shi et al. 2020)	45.50%	54.46%	61.10	56.73	57.46	25.32	34.31
WeakNAS-T (Wu et al. 2021)	45.29%	54.78%	60.70	56.90	57.19	25.41	35.73
Arch-zero (Huang et al. 2022)	45.64%	54.80%	60.21	56.61	57.90	25.73	14.7
Arch-Graph (Huang et al. 2022)	45.81%	**54.90%**	**60.08**	56.58	**58.27**	25.69	**12.2**
LAPT-REA	**45.96%**	**54.89%**	60.18	56.52	57.69	**25.91**	**12.3**
Global Best	46.32%	54.94%	59.38	57.72	59.62	26.27	1

Table 3: Comparison results under same time cost on Trans101 (For Metric: ↑ indicates higher is better, ↓ indicates lower is better, **bold** indicates the best search results). LAPT runs 25 times with different random seeds.

MetaD2A (Lee and Hyung 2021). In LAPT, REA is used for architecture search, and its results (LAPT-REA) are shown in Table 2.

Design principles summarized **by experts:**

P1: The identity layer connecting the input and the output is beneficial to the performance.

P2: Average-pooling layers probably downgrade performance.

P3: Using multiple 3x3 convolution layers in one path improves model performance.

Design principles learned **by our LAPT:**

P1: Introduce skip connections in middle layers to maintain raw features alongside processed ones.

P2: Minimize the use of Zeroize and AvgPool operations, as they seem to be less effective.

P3: Complex transformations appear to perform better than simple ones.

P4: Starting with a convolutional layer helps in learning useful features from the input data.

P5: Utilizing both kernel size 1 and 3 in the convolutional layers allows the model to learn both fine-grained and coarse-grained features.

Figure 3: Design principles separately learned by LAPT and experts. Our LLM-based framework not only learns the identical principles (i.e., $P1 - P3$) with experts but also can reason other valuable principles $P4$ and $P5$.

Compared with NAS methods, our proposed LAPT achieves better search performance and search efficiency by a large margin. Specifically, LAPT can reach the top architectures for CIFAR-10 and CIFAR-100 with 100x less time cost than REA and 10x less time cost than HEBO.

Compared with TNAS methods, benefiting from the search space refinement, LAPT can achieve the SOTA performance. Specifically, all of these methods can reach the top architecture on CIFAR-10, but LAPT is about 25x faster than MetaD2A and 7x faster than TNAS-BO. On CIFAR-100, LAPT is about 12x faster than MetaD2A obviously and is about 7x faster than TNAS-BO.

We further show the general principles learned for the architectures in NAS201 in Figure. 3 and the ones summarized by experts (Yuan et al. 2022). We can see that LAPT can learn the identical principles (i.e., P1-P3) with experts. Most importantly, LAPT can learn two other principles, i.e., P4 and P5. This study demonstrates that our LLM-based approach can automatically learn design principles and produce valuable and varied outcomes.

Comparison on Transfer101. We choose well-known NAS methods and TNAS methods as baselines: (1) the vanilla NAS method REA and the NAS method BONAS (Shi et al. 2020); (2) the state-of-the-art TNAS methods include WeakNAS-T (Wu et al. 2021), Arch-zero and Arch-Graph (Huang et al. 2022). All of these methods have the same search cost, i.e., the number of trained architectures is set to 50. Transferred architectural knowledge of all the TNAS is from the same computation task, i.e., Jigsaw.

From Table 3, our LAPT can achieve better search performance and search efficiency by a large margin than the NAS methods. Specifically, LAPT can beat BONAS on all the 6 tasks. LAPT outperforms REA on 5 out of 6 tasks and achieve the highest average mode rank.

Compared with these TNAS methods, benefiting from the search space refinement, LAPT can achieve the SOTA results. Specifically, LAPT beats WeakNAS-T on most tasks (5 out of 6). Compared with the Arch-zero, LAPT achieves better performance on half of the tasks, i.e., Obj, Auto, and Seg obviously, and achieves comparable performance on other tasks. Compared with the SOTA method Arch-Graph, LAPT can beat it on tasks obj and Seg, and achieve similar performance on other tasks. The learned general principles for Transfer101 are shown in https://arxiv.org/abs/2408.11330.

Comparison on DARTs. In these experiments, LAPT is applied to search for architectures for CIFAR-10 and ImageNet from DARTs. Baselines include: (1) the basic CNNs designed by experts include VGG (Simonyan and Zisserman 2014), ResNet-110 (He et al. 2016), DenseNet-BC (Huang et al. 2017), and SENet (Zagoruyko and Komodakis 2016); (2) the superior NAS methods which have improved state-of-the-art results at that time include SNAS (Xie et al. 2018), and DARTs (Liu, Simonyan, and Yang 2018); (3) SOTA NAS methods include GibbsNAS (Xue et al. 2021), MFE-NAS (Yang et al. 2022), and MASNAS (Dong et al. 2022). In LAPT, the vanilla NAS method REA is used for architecture search, and its search results (LAPT-REA) on CIFAR-10 are shown in Table 4.

From Table 4 we can observe that because of the search

Architecture	Test Acc.(%)	GPU Days
VGG (Simonyan and Zisserman 2014)	93.35	-
ResNet-110 (He et al. 2016)	93.40	-
DenseNet-BC (Huang et al. 2017)	94.81	-
SENet (Zagoruyko and Komodakis 2016)	95.38	-
SNAS (Xie et al. 2018)	97.15	1.5
DARTs (Liu, Simonyan, and Yang 2018)	97.28	4
GibbsNAS (Xue et al. 2021)	97.47	0.5
MSNAS (Dong et al. 2022)	97.32	0.25
MFENAS (Yang et al. 2022)	97.61	0.6
LAPT-REA	**97.35**	**0.1**

Table 4: Performance comparison on CIFAR-10 using the search space DARTs.

Architecture	Top1 Acc.(%)	Params.	GPU Days
InceptionV1	69.8	6.6M	-
MobileNet	70.6	4.2M	-
ShuffleNet	73.7	5.0M	-
PC-DARTs (Xu et al. 2020)	75.8	5.3M	3.8
FairDARTs (Chu et al. 2020)	75.6	4.3M	3.0
Shapley-NAS (Xiao et al. 2022)	76.1	5.4M	4.2
LAPT-REA	**75.1**	**4.6M**	**2.0**

Table 5: Performance comparison on ImageNet using the search space DARTs.

space optimization, LAPT can find the superior architectures more efficiently. Specifically, LAPT achieves superior performance in terms of test accuracy against these handcrafted CNNs including VGG-16, ResNet-110, DenseNet-BC, and SENet on CIFAR-10. Compared to superior NAS methods, LAPT can outperform them. To be specific, LAPT can beat SNAS using 15× fewer GPU days and cost 10× fewer GPU days to achieve better performance than DARTs. Although the test accuracy of LAPT is slightly worse than state-of-the-art methods, it has less time cost. Specifically, LAPT achieves comparable performance on CIFAR-10 with GibbsNAS, MSNAS, and MFENAS but costs 5×, 2.5×, and 6× fewer GPU days.

Search results on ImageNet are shown in Table 5. From this table, we can see that because of the search space refinement, LAPT can achieve similar search results with other NAS methods but with about 1.5-2× less search cost (2 GPU days), which is more time-efficient.

Ablation study

In the proposed LAPT framework, two core components are emphasized: knowledge transfer and principle adaptation. To showcase their efficacy, two different verions of LAPT are performed on Trans101. In the first version, the principle adaptation step is omitted, referred to as **WO Adaptation**. In this scenario, the search space is refined solely based on the design principles derived from previous tasks and never refined during the search process. In the second version, denoted as **WO Transfer**, architectural knowledge from prior tasks is unavailable, and LAPT refines the search space solely based on the architectures designed for the new task. The model ranks of the discovered architectures via these

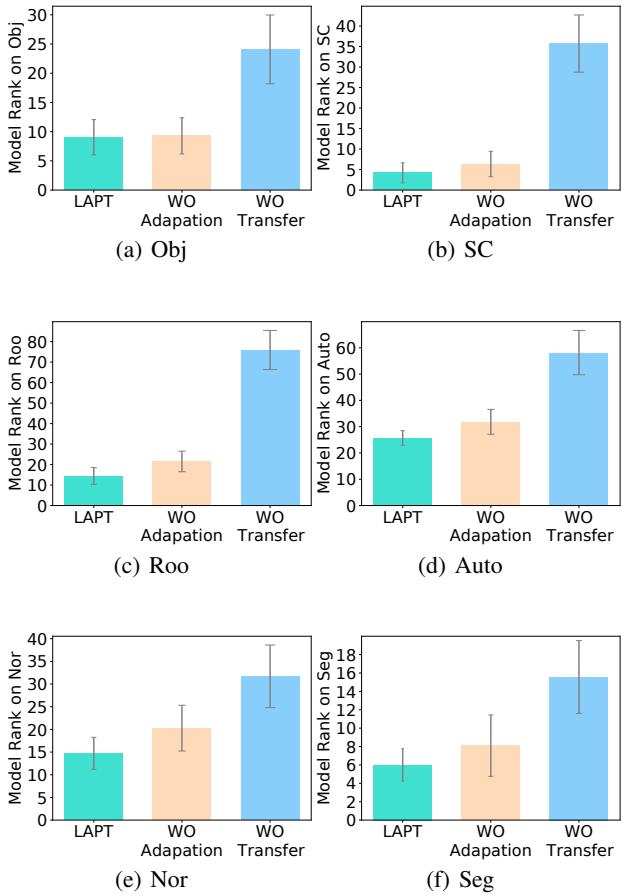

Figure 4: Model rank of different LAPT versions on Trans101.

two versions illustrated in Fig. 4. These results demonstrate that knowledge transfer closely aligns with enhanced search performance. Without employing the transfer strategy, identifying top-ranking architectures becomes challenging. Furthermore, the principle adaptation strategy is more related to convergence of the search method (leading to the lower variance of model ranks).

Conclusion

This paper introduces a novel idea of design principle transfer in NAS, aiming to improve the architecture search efficiency for new tasks. To this end, an LLM-assist framework is proposed to help learn the design principles from established architectures and transfer them well to the new NAS tasks. Experiments on different search spaces have demonstrated that the quality of the refined search spaces is higher than the original ones. Even the vanilla search method can achieve SOTA results when searching in such space. However, in this work, these tasks need to share the high similarity and the same search space. Knowledge transfer across different search spaces and domains should be explored in the future.

Acknowledgments

This work was supported in part by National Key R&D Program of China (2022YFC3801700); in part by the Research Grants Council of the Hong Kong SAR (Grant No. PolyU11211521, PolyU15218622, PolyU15215623, and C5052-23G), and the National Natural Science Foundation of China (Grant No. U21A20512).

References

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

Chen, L.; Xu, F.; Li, N.; Han, Z.; Wang, M.; Li, Y.; and Hui, P. 2024. Large Language Model-driven Meta-structure Discovery in Heterogeneous Information Network. *arXiv preprint arXiv:2402.11518*.

Chu, X.; Zhou, T.; Zhang, B.; and Li, J. 2020. Fair darts: Eliminating unfair advantages in differentiable architecture search. In *European conference on computer vision*, 465–480. Springer.

Cowen-Rivers, A. I.; Lyu, W.; Tutunov, R.; Wang, Z.; Grosnit, A.; Griffiths, R. R.; Maraval, A. M.; Jianye, H.; Wang, J.; Peters, J.; et al. 2022. Hebo: Pushing the limits of sample-efficient hyper-parameter optimisation. *Journal of Artificial Intelligence Research*, 74: 1269–1349.

Dong, H.; Gao, Y.; Wang, H.; Yang, H.; and Zhang, P. 2023. Heterogeneous Graph Neural Architecture Search with GPT-4. *arXiv preprint arXiv:2312.08680*.

Dong, J.; Hou, B.; Feng, L.; Tang, H.; Tan, K. C.; and Ong, Y.-S. 2022. A cell-based fast memetic algorithm for automated convolutional neural architecture design. *IEEE Transactions on Neural Networks and Learning Systems*. Early access, DOI: 10.1109/TNNLS.2022.3155230.

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

Duan, Y.; Chen, X.; Xu, H.; Chen, Z.; Liang, X.; Zhang, T.; and Li, Z. 2021. Transnas-bench-101: Improving transferability and generalizability of cross-task neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5251–5260.

Elsken, T.; Staffler, B.; Metzen, J. H.; and Hutter, F. 2020. Meta-learning of neural architectures for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12365–12375.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4700–4708.

Huang, M.; Huang, Z.; Li, C.; Chen, X.; Xu, H.; Li, Z.; and Liang, X. 2022. Arch-Graph: Acyclic Architecture Relation Predictor for Task-Transferable Neural Architecture Search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11881–11891.

Imani, S.; Du, L.; and Shrivastava, H. 2023. Mathprompter: Mathematical reasoning using large language models. *arXiv preprint arXiv:2303.05398*.

Jawahar, G.; Abdul-Mageed, M.; Lakshmanan, L. V.; and Ding, D. 2023. LLM performance predictors are good initializers for architecture search. *arXiv preprint arXiv:2310.16712*.

Lee, H.; and Hyung, E. 2021. Rapid neural architecture search by learning to generate graphs from datasets. *arXiv preprint arXiv:2107.00860*.

Li, J.-Y.; Zhan, Z.-H.; Tan, K. C.; and Zhang, J. 2021. A meta-knowledge transfer-based differential evolution for multitask optimization. *IEEE Transactions on Evolutionary Computation*, 26(4): 719–734.

Liu, F.; Xialiang, T.; Yuan, M.; Lin, X.; Luo, F.; Wang, Z.; Lu, Z.; and Zhang, Q. 2024. Evolution of Heuristics: Towards Efficient Automatic Algorithm Design Using Large Language Model. In *Forty-first International Conference on Machine Learning*.

Liu, H.; Simonyan, K.; and Yang, Y. 2018. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

Lu, Z.; Sreekumar, G.; Goodman, E.; Banzhaf, W.; Deb, K.; and Boddeti, V. N. 2021. Neural architecture transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(9): 2971–2989.

Lu, Z.; Whalen, I.; Dhebar, Y.; Deb, K.; Goodman, E. D.; Banzhaf, W.; and Boddeti, V. N. 2020. Multiobjective Evolutionary Design of Deep Convolutional Neural Networks for Image Classification. *IEEE Transactions on Evolutionary Computation*, 25(2): 277–291.

Nasir, M. U.; Earle, S.; Togelius, J.; James, S.; and Cleghorn, C. 2023. Llmatic: Neural architecture search via large language models and quality-diversity optimization. *arXiv preprint arXiv:2306.01102*.

Qin, R.; Hu, Y.; Yan, Z.; Xiong, J.; Abbasi, A.; and Shi, Y. 2024. FL-NAS: Towards Fairness of NAS for Resource Constrained Devices via Large Language Models. *arXiv preprint arXiv:2402.06696*.

Radosavovic, I.; Kosaraju, R. P.; Girshick, R.; He, K.; and Dollár, P. 2020. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10428–10436.

Real, E.; Aggarwal, A.; Huang, Y.; and Le, Q. V. 2019. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 4780–4789.

Shala, G.; Elsken, T.; Hutter, F.; and Grabocka, J. 2023. Transfer NAS with meta-learned bayesian surrogates. In *The Eleventh International Conference on Learning Representations*.

Shi, H.; Pi, R.; Xu, H.; Li, Z.; Kwok, J.; and Zhang, T. 2020. Bridging the gap between sample-based and one-shot neural architecture search with bonas. In *Advances in Neural Information Processing Systems*, 1808–1819.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Wang, H.; Gao, Y.; Zheng, X.; Zhang, P.; Chen, H.; and Bu, J. 2023. Graph neural architecture search with gpt-4. *arXiv preprint arXiv:2310.01436*.

Wu, J.; Dai, X.; Chen, D.; Chen, Y.; Liu, M.; Yu, Y.; Wang, Z.; Liu, Z.; Chen, M.; and Yuan, L. 2021. Stronger nas with weaker predictors. In *Advances in Neural Information Processing Systems*, 28904–28918.

Wu, X.; Wu, S.-h.; Wu, J.; Feng, L.; and Tan, K. C. 2024. Evolutionary Computation in the Era of Large Language Model: Survey and Roadmap. *IEEE Transactions on Evolutionary Computation*. Early access, DOI: 10.1109/TEVC.2024.3506731.

Xiao, H.; Wang, Z.; Zhu, Z.; Zhou, J.; and Lu, J. 2022. Shapley-NAS: Discovering operation contribution for neural architecture search. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 11892–11901.

Xie, S.; Zheng, H.; Liu, C.; and Lin, L. 2018. SNAS: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*.

Xu, Y.; Xie, L.; Zhang, X.; Chen, X.; Qi, G.-J.; Tian, Q.; and Xiong, H. 2020. PC-DARTs: Partial channel connections for memory-efficient differentiable architecture search. In *International Conference on Learning Representations*.

Xue, C.; Wang, X.; Yan, J.; Hu, Y.; Yang, X.; and Sun, K. 2021. Rethinking Bi-Level Optimization in Neural Architecture Search: A Gibbs Sampling Perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 10551–10559.

Yang, S.; Tian, Y.; Xiang, X.; Peng, S.; and Zhang, X. 2022. Accelerating Evolutionary Neural Architecture Search via Multifidelity Evaluation. *IEEE Transactions on Cognitive and Developmental Systems*, 14(4): 1778–1792.

Yuan, J.; Liu, M.; Tian, F.; and Liu, S. 2022. Visual analysis of neural architecture spaces for summarizing design principles. *IEEE Transactions on Visualization and Computer Graphics*, 29(1): 288–298.

Zagoruyko, S.; and Komodakis, N. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146*.

Zhang, S.; Gong, C.; Wu, L.; Liu, X.; and Zhou, M. 2023. Automl-gpt: Automatic machine learning with gpt. *arXiv preprint arXiv:2305.02499*.

Zheng, M.; Su, X.; You, S.; Wang, F.; Qian, C.; Xu, C.; and Albanie, S. 2023. Can GPT-4 perform neural architecture search? *arXiv preprint arXiv:2304.10970*.

Zhou, X.; Qin, A.; Sun, Y.; and Tan, K. C. 2021a. A Survey of Advances in Evolutionary Neural Architecture Search. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, 950–957.

Zhou, X.; Qin, A. K.; Gong, M.; and Tan, K. C. 2021b. A survey on evolutionary construction of deep neural networks. *IEEE Transactions on Evolutionary Computation*, 25(5): 894–912.

Zhou, X.; Wang, Z.; Feng, L.; Liu, S.; Wong, K.-C.; and Tan, K. C. 2023. Towards Evolutionary Multi-Task Convolutional Neural Architecture Search. *IEEE Transactions on Evolutionary Computation*, 28(3): 682–695.