Evolutionary Transfer Neural Architecture Search across Spaces via Representation Learning

Boyu Hou, Liang Feng, *Senior Member, IEEE,* Xuefeng Chen, *Member, IEEE,* Jing Tang, *Member, IEEE,* Kay Chen Tan, *Fellow, IEEE,* and Xiaofeng Liao, *Fellow, IEEE*

***Abstract*—Neural Architecture Search (NAS) has emerged as a crucial method for automating the design of deep learning models. Despite its potential, NAS frequently requires substantial computational and hardware resources. To mitigate these challenges, transferable NAS (TNAS) has been introduced, leveraging prior NAS results to enhance performance on new tasks. However, existing methods largely focus on knowledge transfer within identical neural search spaces, overlooking the potential for cross-domain transferability. Motivated by this gap, we explore evolutionary TNAS across heterogeneous search spaces by learning common neural representations. In particular, we introduce a novel approach that encodes both operational and topological information of neural architectures into a unified sequence using a simple tokenizer. This sequence is then processed by a variational auto-encoder, with a Transformer-based encoder to capture rich neural representations and a decoder that reconstructs the original sequence. By utilizing these latent representations, we further establish an inter-domain mapping that acts as a bridge, enabling effective explicit solution transfer among diverse search spaces to enhance the evolutionary NAS process. To harness this capability, we develop an evolutionary sequential transfer optimization approach that transfers knowledge during population initialization, providing both flexibility and adaptability. To the best of our knowledge, this work serves as the first attempt in the literature exploring evolutionary TNAS across diverse spaces. Moreover, we demonstrate the utility of our method through comprehensive empirical studies using different architecture spaces, including NAS-Bench-101, NAS-Bench-201, and the DARTS search space. Our results show that the proposed method significantly enhances the adaptability and performance of NAS across varied domains.**

***Index Terms*—neural architecture search, transfer learning, representation learning**

I. Introduction

DEEP learning has shown success in various fields [1]–[3]. However, developing effective deep neural networks (DNNs) remains a challenging and time-intensive process, necessitating substantial domain expertise and experimental validation. This challenge has spurred the development of Neural Architecture Search (NAS), a technique that automates the discovery of optimal DNN architectures tailored to specific tasks and datasets.

Traditionally, the design of DNNs relied heavily on manual tuning and expert intuition, which are both resource-intensive and potentially biased towards the designer's experience. NAS introduces a systematic, data-driven approach that significantly reduces human labor and subjectivity in the architectural design process [4], [5]. However, this automation comes with its own set of challenges, primarily the high computational cost [6]. Evaluating numerous architectures to determine the most effective one for a particular task demands substantial computational resources, which can be a limiting factor in research and practical applications. Consequently, a considerable amount of works [6]–[10] have been dedicated to finding efficient methods for rapidly searching of high-performing neural architectures in specific problem scenarios.

Transferable NAS (TNAS) has emerged as a promising strategy for alleviating the substantial resource requirements of NAS. By leveraging architectural patterns discovered in one domain and applying them to related tasks, TNAS can streamline the optimization process of neural architectures [11]. This efficiency is achieved by introducing bias into the search process, informed by insights gained from previously solved NAS tasks. Evolutionary TNAS further enhances this concept by employing evolutionary algorithms to introduce flexibility and mitigate negative transfer, resulting in notable advancement [12], [13]. However, evolutionary TNAS is limited by search space consistency, restricting its adaptability. Search spaces vary significantly in complexity and constraints [5], [7], [8], [14], affecting knowledge transfer across domains. This heterogeneity necessitates restarting the search process for new spaces, reducing efficiency and rendering prior knowledge less applicable.

Addressing the issue of cross-domain transferability in NAS is critical for overcoming these obstacles while the potential for transferring NAS solutions across diverse and heterogeneous search spaces remains largely unexplored and underutilized. One notable effort is *Cross-Domain Predictor* (CDP) [15], which leverages a domain adaptation approach to construct a neural performance predictor across different search spaces. However, CDP falls short in enabling direct knowledge transfer during the search process and is constrained by the requirement for similarity between search spaces.

In this paper, we introduce Bridge (**B**uilding **R**epresentation for **I**nter-**D**omain Hetero**g**eneous **E**volutionary TNAS), a novel approach that leverages neural representation learning to facilitate cross-domain knowledge transfer in evolutionary NAS.

Corresponding Authors: Liang Feng and Xuefeng Chen.

Boyu Hou, Liang Feng, Xuefeng Chen and Xiaofeng Liao are with the College of Computer Science, Chongqing University; and the Chongqing Key Laboratory of Big Data Intelligence and Privacy Computing, Chongqing 400044, China (e-mail: {byhou,lfeng,xfchen,xfliao}@cqu.edu.cn).

Jing Tang is with the Data Science and Analytics Thrust, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou 511453, China (e-mail: jingtang@ust.hk).

Kay Chen Tan is with the Department of Data Science and Artificial Intelligence, The Hong Kong Polytechnic University, Kowloon 999077, Hong Kong SAR (e-mail: kctan@polyu.edu.hk).

The core ingredients of BRIDGE contains a neural representation learning framework designed to construct a latent space that encodes comprehensive neural architecture information. This representation is further enriched with architecture-specific performance metrics, enabling the establishment of effective cross-domain mappings for solution transfer. To achieve this, we design a representation learner based on a variational auto-encoder (VAE), incorporating a Transformer-based encoder to encode architectural sequences into a latent representation space. By leveraging these latent representations, BRIDGE learns an inter-domain mapping, effectively bridging the gap between heterogeneous search spaces and facilitating explicit solution transfer. This method enables efficient utilization of insights gathered from diverse neural architecture spaces, thereby improving both the efficiency and efficacy of new NAS endeavors. Another key components in the proposed BRIDGE is an Evolutionary Sequential Transfer Optimization (ESTO) [16]–[18] search algorithm which enables a transferable neural architecture search process across spaces using the learned inter-domain mapping. This approach not only expands the applicability of NAS but also harnesses cumulative knowledge from multiple search spaces, streamlining the architectural search process and reducing computational overhead. Empirical evaluations conducted across search spaces of varying scales and complexities demonstrate that BRIDGE significantly outperforms baseline methods lacking solution transfer mechanisms. These results validate the effectiveness of the proposed BRIDGE in enhancing NAS outcomes, underscoring the potential of our approach to advance the field of NAS. The main contributions of this paper can be summarized as:

- To the best of our knowledge, this work present the first explicit solution transfer approach for evolutionary TNAS across heterogeneous domains, addressing a key challenge in NAS research.
- We propose a neural representation learning method integrating Transformer and VAE architectures to establish effective cross-domain mappings and optimize latent space representations with architecture performance metrics.
- We introduce an ESTO-based cross-domain evolutionary TNAS approach, leveraging unified encoding strategies and evolutionary operators to enhance knowledge transfer. Empirical results confirm substantial performance improvements.

The rest of this paper is structured as follows: Section II provides essential background information and reviews related work. Section III presents the theoretical foundations and technical details of our proposed approach, i.e., BRIDGE. Section IV details the experimental setup and provides a comprehensive analysis of our findings. Finally, Section VI summarizes the contributions of this work and discusses potential future directions.

II. PRELIMINARY

This section presents essential background on NAS and reviews recent advancements in two related areas critical to our work: knowledge transfer in NAS and representation learning for neural architectures. These topics provide the foundational context for our proposed approach and situate our contributions within the current landscape of NAS research.

A. Neural Architecture Search

Neural Architecture Search (NAS) has evolved as a pivotal technique for automating the design of deep learning models. It systematically identifies optimal architectures tailored to specific tasks, thus mitigating the reliance on expert intuition and manual tuning. Formally, NAS can be formulated as a bilevel optimization problem. Let $\mathcal{A}$ represent the search space of neural architectures and $\mathcal{W}$ denote the space of weights associated with a specific architecture. The objective of NAS is to find an optimal architecture $a^* \in \mathcal{A}$ that minimizes a validation loss $\mathcal{L}_{\text{valid}}$, which depends on the architecture a and its corresponding optimal weights $w^*(a)$:

$$\begin{aligned} a^* &= \arg\min_{a \in \mathcal{A}} \mathcal{L}_{\text{valid}}(w^*(a), a) \\ w^*(a) &= \arg\min_{w \in \mathcal{W}} \mathcal{L}_{\text{train}}(w, a) \end{aligned} \tag{1}$$

where the optimal weights $w^*(a)$ for a given architecture a are often obtained by minimizing the training loss $\mathcal{L}_{\text{train}}$.

NAS frameworks evaluate myriad architectural configurations to determine the most effective designs, which traditionally entails a high computational cost and extensive resource allocation. Early approaches, such as Reinforcement Learning (RL) [4], [5], [19]–[21] and Evolutionary Algorithms (EA) [8], [10], [22]–[24], demonstrated the feasibility of automating architecture search but were often criticized for their resource-intensive nature. Subsequently, techniques such as parameter sharing, one-shot models, and weight inheritance emerged to reduce the computational overhead by reusing model weights across different architecture evaluations. More recent advancements, such as DARTS [7] and NAO [25], have leveraged gradient-based optimization and latent space representations to further reduce computational overhead, making NAS more practical for large-scale applications.

Despite these advances, the computational cost of NAS remains a significant obstacle. Researchers have explored numerous efficiency-boosting techniques, including the use of surrogate models to predict the performance of candidate architectures, thereby minimizing the need for exhaustive evaluations.

B. Knowledge Transfer in NAS

Neural Architecture Search (NAS) faces significant computational challenges that researchers have addressed through knowledge transfer mechanisms to accelerate the search process without compromising solution quality. Initial approaches focused on intra-task knowledge transfer using supernets, which represent candidate architectures as subgraphs with shared parameters [5], [6], [26]. This parameter sharing enables efficient architecture evaluation with minimal training. Research subsequently expanded to more complex transfer scenarios between different tasks within the same search space. While straightforward approaches transfer NAS solutions from source to target tasks [5], [7], this method requires high task similarity

and often yields suboptimal performance with significant domain shifts [7]. To address this limitation, methods like Arch-Graph [27] predict task-specific architectures using task embeddings, while EMT-NAS [12] and MTNAS [13] facilitate evolutionary knowledge transfer across tasks.

Despite these advances, fixed search spaces fundamentally limit the broad applicability of knowledge transfer methodologies. Real-world computational tasks and deployment environments exhibit inherent heterogeneity, necessitating diverse search spaces optimized for specific resource constraints and performance objectives. This heterogeneity manifests in two critical scenarios: (1) translating architectures from high-performance computing environments to resource-constrained edge devices, and (2) adapting architectural solutions across disparate application domains, such as from image classification to specialized tasks in object detection or medical image analysis. These challenges underscore the necessity for robust cross-domain transfer capabilities in NAS. Such capabilities would enhance both methodological flexibility and generalizability while potentially improving search efficiency through systematic knowledge leverage across heterogeneous domains. The fundamental challenge lies in establishing effective mapping mechanisms between search spaces with distinct operational characteristics and topological configurations.

Recent investigations have approached this challenge through search space alignment methodologies. For instance, CDP [15] indirectly facilitates knowledge transfer between different search spaces by employing domain adaptation, thereby training a cross-domain neural architecture performance predictor. *A General-Purpose Transferable Predictor for NAS* [28] also tried to transfer the predictor by representing any given candidate convolutional neural network with a computation graph that consists of primitive operators.

However, to the best of our knowledge, there is currently a lack of comprehensive research focused on the explicit transfer of solutions across domains in NAS, which would facilitate the exploration of diverse search spaces, enabling efficient neural architecture designing tailored to various computing resource constraints and deployment scenarios. Our work advances beyond traditional TNAS methods by building bridges for explicit solution transfer across domains by learning representations of neural architectures and integrating population-based search strategies, enabling more flexible, scalable, and effective transfers.

C. Representation Learning of Neural Architectures

To improve the efficiency of NAS, numerous studies have concentrated on representation learning to model neural architectures, thereby constructing a latent representation space that captures their essential characteristics. Performance predictors have been developed to operate within this latent space, allowing for efficient estimation of the quality of neural architectures. Early predictors utilize models such as LSTM [25], [29], [30], MLP [31], [32] and Random Forest [33] to accelerate the evaluating of architecture. More recently, there has been a focus on utilizing GCNs [34]–[38] to model the graph-formatted data (e.g. adjacency matrix and node features) of architectures. The *Neural Predictor* [37] encodes each node in the architecture by calculating the mean of two modified GCN layers, which use the adjacency matrix and the transpose of the adjacency matrix to propagate information, respectively. The Transformer [39] also serves as a neural architecture representation learner and has found widespread application in numerous recent works [40], [41]. The self-attention mechanism in Transformers is particularly advantageous for modeling intricate dependencies and interactions between different architectural components, such as topologies and operators. NAR-Former [41] encodes both operational and topological information of a neural network into a single sequence, leveraging the Transformer to generate a concise vector representation that facilitates the prediction of key metrics, such as latency and accuracy, for both individual cell architectures and complete neural networks. The findings of these studies further demonstrate that neural architectures can be effectively clustered within the representation space according to their performance.

Additionally, recent research efforts, such as those by NAO, etc. [25], [38], [42] have shown the feasibility of conducting searches within the space of representations. They rapidly search through continuous optimization methods like Bayesian Optimization (BO) and Gradient Descent (GD) by constructing a continuous representation space. Specifically, these continuous representations allow for smoother navigation of the architecture space [38], enabling more efficient exploration and optimization of neural network designs.

The success of both performance predictor-based methods and continuous search approaches highlights the potential for establishing transfer mappings between distinct domains using representation learning. By leveraging the distributional characteristics of these continuous spaces, we hypothesize that meaningful connections can be formed across different architecture search spaces, even when their original formulations vary significantly. In this study, we introduce a novel approach whereby neural architecture representations serve as a conduit for cross-domain TNAS, enabling the exploitation of knowledge derived from disparate search spaces.

III. Proposed Method

This section presents a comprehensive description of our proposed cross-domain evolutionary transfer neural architecture search framework Bridge. We begin with an overview of the entire framework, followed by in-depth discussions of its key components. These components include building neural architecture representation, constructing cross-domain transfer mappings, and performing cross-domain evolutionary transfer for NAS. Each of these elements plays a crucial role in our proposed approach to TNAS across heterogeneous search spaces.

A. Overview of Bridge

As shown in Fig. 1, the entire framework of Bridge can be decomposed into several steps. Initially, the neural architecture representation learner undergoes training on both the source and target domains independently. This process aims to establish the encoding and decoding capabilities necessary for bridging the

This article has been accepted for publication in IEEE Transactions on Evolutionary Computation. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2025.3565326

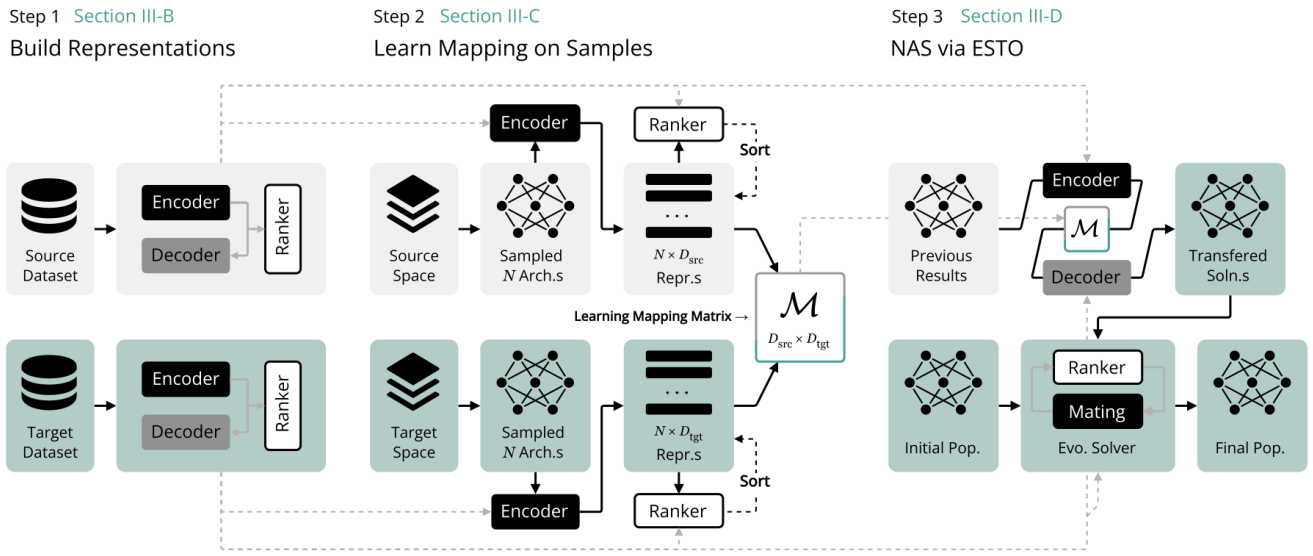


Fig. 1. Overview of the BRIDGE framework. The whole process can be decomposed to three steps: (1) building representations with supervised and unsupervised representation learning; (2) learning mapping between representations from source and target domains; (3) transfer historical solutions from source domain with representation model and mapping to target domain.

architecture space and the representation space. Subsequently, using performance-aligned sampling in the representation space, the model learns the mapping between source and target domains. The resulting components, including the constructed encoders, decoders, performance predictors, and representation mappings, play pivotal roles in facilitating the transfer of historical solutions across domains.

B. Building Representation of Neural Architecture

1) Neural Architecture Tokenizer: Neural architectures are fundamentally computational graphs, where data undergoes transformations through operators in the architecture, adhering to the flow of the graph and interacting with other data streams or progressing to subsequent operators. Consequently, neural architectures can be inherently conceptualized as a form of graph-structured data, or directed acyclic graph (DAG) specifically.

To accommodate a diverse array of heterogeneous search spaces, we undertake a targeted design for the two primary cell-based neural architecture encoding paradigms, namely, "Operation On Node" (OON) and "Operation On Edge" (OOE). This tailored approach ensures adaptability across various architectural structures, allowing the tokenizer to effectively capture and represent the intricacies of different neural architectures.

OON Encoding. In the "Operation On Node" encoding paradigm, as exemplified by NASNet [5] and NAS-Bench-101 [43], operators are treated as nodes, and data streams are considered edges. This coding methodology involves the separation of topology information and operator information into proximity matrices and node labels, respectively. Formally, given an OON encoded neural architecture cell $\boldsymbol{x} = (A, \boldsymbol{ops})$ with n nodes, where A is the adjacency matrix while $\boldsymbol{ops}$ is the sequence of operation tokens, the proposed neural architecture

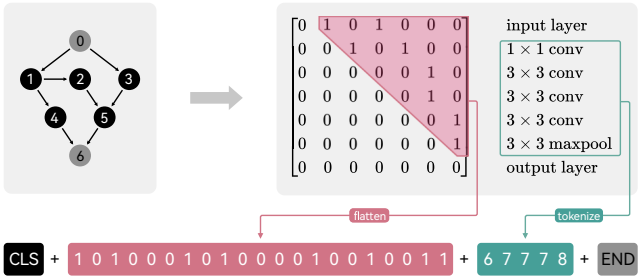


Fig. 2. Illustration of tokenizing an OON architecture (from NAS-Bench-101) as a sequence. The symbol "+" represents concatenating two sequences.

tokenizer will encode it as a sequence $\mathrm{Tkn}_{\mathrm{N}}(\boldsymbol{x})$ with the length of $l_{\mathrm{N}} + 2$:

$$\begin{cases} \mathrm{Tkn}_{\mathrm{N}}(\boldsymbol{x}) = \{\texttt{CLS}\} \cup \underbrace{\mathrm{Triu}(A) \cup \boldsymbol{ops}_{2:n-1}}_{l_{\mathrm{N}} \text{ tokens}} \cup \{\texttt{END}\} \\ l_{\mathrm{N}} = \dfrac{n(n-1)}{2} + (n-2) = \dfrac{1}{2}(n^2 + n - 4) \end{cases} \tag{2}$$

where we use special tokens `CLS` and `END` to indicate the start and end of a sequence, respectively. The upper triangular array of a given adjacency matrix is represented by $\mathrm{Triu}(\cdot)$. Fig. 2 illustrates the OON encoding: the sequence combines flattened upper-triangular adjacency matrix elements ($n(n-1)/2$ tokens) and non-input/output operator node labels ($n-2$ tokens), mapped via a vocabulary (Fig. 3).

OOE Encoding. In contrast to the OON encoding, the "Operator On Edge" encoding paradigm treats operators as edges and data as nodes. This coding approach, exemplified by search spaces like that in DARTS [7] and NAS-Bench-201 [44], facilitates the compression of both topological and operator information of the neural architecture into the adjacency matrix, leading to denser information representation regarding the

NAS-Bench-101

0	unlink
1	linked
2	conv11_bn_relu
3	conv33_bn_relu
4	maxpool33
5	input
6	output

NAS-Bench-201

0	unlink
1	skip_connect
2	nor_conv_11
3	nor_conv_33
4	avg_pool_33

DARTS

0	unlink
1	skip_connect
2	max_pool_33
3	avg_pool_33
4	sep_conv_33
5	sep_conv_55
6	dil_conv_33
7	dil_conv_55

Fig. 3. The operator vocabularies for NAS-Bench-101, NAS-Bench-201 and DARTS space respectively.

relationships between operators and their interactions with data nodes. A neural architecture cell $\boldsymbol{x} = (A)$ containing n nodes will be coded as a sequence $\mathrm{Tkn}_{\mathrm{E}}(\boldsymbol{x})$ of length $l_{\mathrm{E}} + 2$:

$$\begin{cases} \mathrm{Tkn}_{\mathrm{E}}(\boldsymbol{x}) = \{\texttt{CLS}\} \cup \underbrace{\mathrm{Triu}(A)}_{l_{\mathrm{E}}\text{ tokens}} \cup \{\texttt{END}\} \\ l_{\mathrm{E}} = \dfrac{n(n-1)}{2} = \dfrac{1}{2}(n^2 - n) \end{cases} \tag{3}$$

where the sequence is constructed by flattening the upper triangular arrays of adjacency matrices along with the inclusion of special tokens. For architectures with multiple cell types (e.g., DARTS), their token sequences are concatenated for a holistic representation.

2) Transformer-based Variational Auto-Encoder: We develop a Transformer-based representation learner [39], [41] due to its suitability for handling variable-length architecture sequences, and capturing global structural and semantic interactions, enabling robust and efficient representation.

The representation learner is built upon the Variational Auto-Encoder (VAE) framework (Fig. 4), inspired by previous work in bi-directional architecture mapping [38], [42]. VAE offers key advantages, including injective encoding, which ensures unique local structural information retention, facilitating distinct representations for each architecture. Additionally, VAEs naturally cluster similar architectures in the latent space, enabling smooth transitions beneficial for performance prediction [22].

Initially, the VAE encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ parameterized by ϕ is employed to map the input data $\boldsymbol{x}$, which consists of a finite number of independent identical distribution samples from an unknown distribution, to a continuous latent variable $\boldsymbol{z}$. Following this encoding step, a probabilistic generative model $p_\theta(\boldsymbol{x}|\boldsymbol{z})$, as known as the decoder, reconstructs the latent variables $\boldsymbol{z}$ back to their original representation. The parameters ϕ and θ of the encoder and decoder, respectively, are optimized by maximizing

$$\mathcal{L}_{\phi,\theta}(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}\big[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\big] - D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p_\theta(\boldsymbol{z})\big) \tag{4}$$

where the first term is the *Evidence Lower Bound* (ELBO), which represents the reconstruction loss, which ensures a high similarity between the input data and the data generated by the decoder and plays a crucial role in maintaining fidelity between the original and reconstructed data. In contrast, the second term is the *Kullback-Leibler Divergence* (KLD), serving as a regularization term for the latent space, which encourages the distribution of latent variables to remain close to the prior distribution, thus promoting a well-behaved and structured latent space.

The following outlines the specifications of the proposed Transformer-based VAE encoder and decoder models that are designed to effectively capture and reconstruct the complex structural relationships within neural architectures.

Encoder. For a given neural architecture $\boldsymbol{x}$, it undergoes several transformations before input into the Transformer encoder. Initially, the architecture is tokenized into a sequence of discrete tokens using Eq. (2) or Eq. (3), each representing a specific architectural component or operation, to ensure the compatibility of the input with the Transformer encoder's format. Subsequently, each token is mapped to a high-dimensional embedding vector using a learned embedding layer $\mathrm{Emd}(\cdot)$, capturing semantic relationships between architectural components. Positional embeddings E_{pos} are then added to provide positional information, crucial for the Transformer Encoder to understand the order of architectural components, as illustrated in Eq. (5).

The resulting sequence of token embeddings, enriched with positional information, is fed into the Transformer encoder. The input undergoes processing through multi-head self-attention layer $\mathrm{MSA}(\cdot)$ and feed-forward layer $\mathrm{FF}(\cdot)$, yielding a contextualized representation of the neural architecture. Layer normalization $\mathrm{LN}(\cdot)$ is applied to the inputs of each layer, and the outputs are added to the residual connections, following the typical Transformer layer design. Formally, an architecture $\boldsymbol{x}$ can be encoded using an N-layers Transformer-based architecture encoder $\mathrm{TAE}(\cdot)$ as

$$\mathrm{TAE}(\boldsymbol{x}) \Rightarrow \begin{cases} \boldsymbol{h}^{(0)} = \mathrm{Emd}(\mathrm{Tkn}(\boldsymbol{x})) + E_{\mathrm{pos}} \\ \boldsymbol{h}^{(i)}_{\mathrm{att}} = \mathrm{LN}(\mathrm{MSA}(\boldsymbol{h}^{(i-1)})) + \boldsymbol{h}^{(i-1)} \\ \boldsymbol{h}^{(i)} = \mathrm{LN}(\mathrm{FF}(\boldsymbol{h}^{(i)}_{\mathrm{att}})) + \boldsymbol{h}^{(i)}_{\mathrm{att}} \end{cases} \tag{5}$$

where $i \in [1, N]$, $\boldsymbol{h}^{(i)}$ and $\boldsymbol{h}^{(i)}_{\mathrm{att}}$ represent the hidden states and attention states of the i-th layer, respectively.

Specifically, the hidden states of the `CLS` token are utilized as the memory of the input architecture sequence, and thus we can compute two representations

$$\boldsymbol{m} = \boldsymbol{h}^{(N)}[1, :] \tag{6}$$

$$\boldsymbol{r}_\mu = \mathrm{FC}_\mu(\boldsymbol{m}) \tag{7}$$

$$\boldsymbol{r}_\sigma = \mathrm{FC}_\sigma(\boldsymbol{m}) \tag{8}$$

where $\boldsymbol{m}$ represents the memory of the input architecture, while $\boldsymbol{r}_\mu$ and $\boldsymbol{r}_\sigma$ are the mean and variance of the distribution of learned neural architecture representations, respectively. The two $\mathrm{FC}(\cdot)$ represent fully-connected layers in the same settings. Thus, the outputs of our encoder are the parameters of the approximate posterior distribution

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{r}_\mu, \Sigma), \tag{9}$$

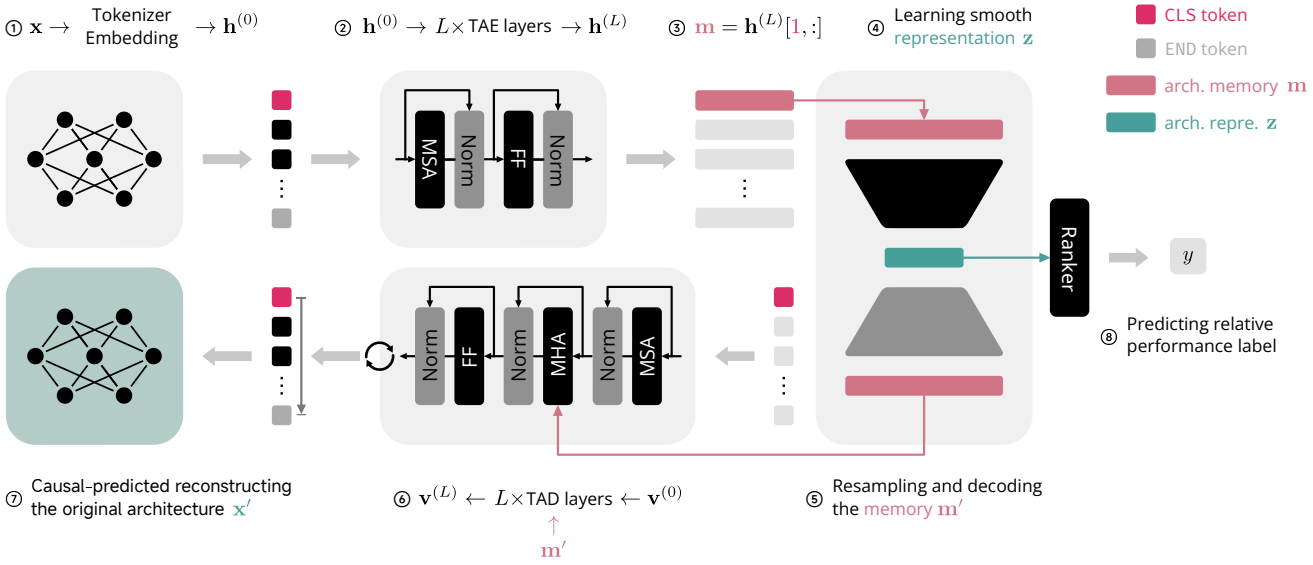


Fig. 4. Illustration of the proposed representation learner. In the pre-training phase, only the neural architecture reconstruction loss is optimized, while the relative performance predictor (Ranker) is not involved in training. In the fine-tuning phase, reconstruction loss and prediction loss are jointly optimized.

Algorithm 1: Architecture Causal Reconstruction

Input: Encoder memory $\mathbf{m}'$, Initial token y_0
Output: Complete output sequence $\boldsymbol{y}$

Initialize $\boldsymbol{y} \leftarrow \{y_0\}$;
for $t \leftarrow 1$ **to** T **do**
  Decoder hidden state $\boldsymbol{v}_t \leftarrow \text{TAD}(\boldsymbol{y})$;
  Compute token probability distribution $p(y_t|\mathbf{h})$;
  Sample token y_t from $p(y_t|\boldsymbol{h})$;
  Append y_t to $\boldsymbol{y}$;
  if y_t = END **then** break;
end for
return $\boldsymbol{y}$;

in which $\Sigma = \boldsymbol{r}_\sigma I$ acts as the variance-covariance matrix of the multi-variate normal distribution.

Decoder. The Transformer-based VAE decoder $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ reconstructs the variable-length architecture $\boldsymbol{x}$ from a latent point $\boldsymbol{z}$. For stability, it uses causal prediction based on reconstructed memory (Algorithm 1). Utilizing a linear layer with activation functions, the reconstructed memory $\boldsymbol{m}' = \text{ReLU}(\text{FC}(\boldsymbol{z}))$ is obtained. The reconstruction process commences with an initial sequence containing a CLS and recursively predicts the next token until the END token is encountered or until the sequence reaches its maximum length L.

$$\text{TAD}(\boldsymbol{y}; \boldsymbol{m}') \Rightarrow \begin{cases} \boldsymbol{v}^{(0)} = \text{Emd}(\text{Tkn}(\boldsymbol{y})) + E_{\text{pos}} \\ \boldsymbol{s}^{(i)} = \text{LN}(\text{MSA}(\boldsymbol{v}^{(i-1)})) + \boldsymbol{v}^{(i-1)} \\ \boldsymbol{c}^{(i)} = \text{LN}(\text{MHA}(\boldsymbol{m}', \boldsymbol{s}^{(i)})) + \boldsymbol{s}^{(i)} \\ \boldsymbol{v}^{(i)} = \text{LN}(\text{FF}(\boldsymbol{c}^{(i)})) + \boldsymbol{c}^{(i)} \end{cases} \tag{10}$$

where $i \in [1, N]$, $\boldsymbol{v}^{(i)}$ represents the hidden states of the i-th layer in the decoder. $\boldsymbol{s}^{(i)}$ and $\boldsymbol{c}^{(i)}$ represent the intermediate states of multi-head self-attention.

3) Incorporate Relative Performance Knowledge: Architecture performance knowledge is incorporated to enable cross-domain mapping via representation spaces, improving transfer and search efficiency. We enhance representations using a performance predictor (ranker) trained on supervised performance signals. The ranker adjusts latent distributions towards performance features, aids mapping construction via efficient labeling, and guides the search.

Following ReNAS [31], a relativistic neural architecture performance predictor is implemented using a multi-layer perceptron with activation functions. By focusing on capturing relative performance differences, the model more effectively differentiates architectural rankings. Given architecture representations $Z = \{\boldsymbol{z}^i\}_1^n$, the relativistic predictor (ranker) is trained by optimizing:

$$\begin{aligned} \Delta_{\text{pred}}^{(i,j)}(Z) &= \psi_\omega(\boldsymbol{z}^i) - \psi_\omega(\boldsymbol{z}^j) \\ \Delta_{\text{sign}}^{(i,j)}(Z) &= \text{Sign}(y_i - y_j) \\ \mathcal{L}_{\text{rel}} &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \exp\left(\Delta_{\text{pred}}^{(i,j)}(Z) \times \Delta_{\text{sign}}^{(i,j)}(Z)\right) \end{aligned} \tag{11}$$

where $\psi_\omega(\cdot)$ is the performance predictor output, $\Delta_{\text{pred}}^{(i,j)}(Z)$ represents the difference in predicted performance, and $\Delta_{\text{sign}}^{(i,j)}(Z)$ measures the relative ranking between the true values of performance.

4) Hybrid Supervised Training: Neural architecture representation construction involves both unsupervised pre-training and supervised fine-tuning, each serving complementary purposes for robust, transferable representations. During pre-training, the representation learner processes architectures from both source and target domains through encoding and decoding, establishing foundational understanding of architectural patterns independent of performance metrics. The subsequent fine-tuning phase incorporates task-specific performance data to optimize representations for the target domain while maintaining transferability, aligning learned representations with domain-specific requirements.

This dual-phase approach minimizes reliance on labeled data — valuable in new domains with limited performance measurements. Unsupervised pre-training enables broad architectural pattern recognition, while fine-tuning ensures domain-specific optimization, balancing knowledge transfer with domain adaptation without compromising representation quality.

Unsupervised Pre-training. The pre-training phase leverages abundant unlabeled data for unsupervised learning, avoiding the prohibitive costs of obtaining performance labels. During this stage, optimization focuses on minimizing reconstruction loss as the model reconstructs architectural configurations from the representation space. Based on Eq. (4), the reconstruction and regularization losses are:

$$\mathcal{L}_{\text{rec}} = \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\right] \tag{12}$$

$$\begin{aligned}\mathcal{L}_{\text{reg}} &= -D_{\text{KL}}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p_\theta(\boldsymbol{z})\big) \\ &= -\frac{1}{2}\sum\left(1 + \log(\boldsymbol{r}_\sigma) - \boldsymbol{r}_\mu^2 - \boldsymbol{r}_\sigma\right)\end{aligned} \tag{13}$$

In the pre-training stage of our framework, a notable feature is the deliberate inactivity of the relative performance predictor, or ranker. This component remains dormant, abstaining from direct involvement in the training process. Our approach employs a strategic decoupling of optimization objectives during this phase. This methodology enables our model to learn generalizable features of neural architectures that are not tethered to domain-specific performance metrics. The decoupling process facilitates the capture of intrinsic architectural characteristics, fostering a more versatile and transferable representation. Consequently, this approach lays a robust foundation for subsequent cross-domain transfer tasks. Formulary, the optimization object can be expressed as:

$$\mathcal{L}_{\text{pt}} = \mathcal{L}_{\text{rec}} + \alpha\mathcal{L}_{\text{reg}} \tag{14}$$

where α is a weight factor that balances the reconstruction and regularization losses.

Supervised Fine-tuning. In the fine-tuning phase, the neural architecture representation learner engages in a process of refinement, harnessing supervised learning techniques. This stage pivots towards a paradigm where labeled data, albeit scarce compared to the vast pool of unlabeled samples, serves as the cornerstone for honing the predictive capabilities of the representation mappings. By joint optimizing reconstruction and relative prediction loss, the finetuning phase yields calibrated representations adept at predicting relative performance, facilitating cross-domain knowledge transfer. This phase culminates in the emergence of finely calibrated neural architecture representations, poised to facilitate seamless knowledge transfer and domain adaptation across heterogeneous environments. The optimization process during this stage is encapsulated in the following objective function:

$$\mathcal{L}_{\text{ft}} = \mathcal{L}_{\text{rec}} + \alpha\mathcal{L}_{\text{reg}} + \beta\mathcal{L}_{\text{rel}} \tag{15}$$

where $\mathcal{L}_{\text{rel}}$ is the relative performance prediction loss, while hyper-parameters α and β control the balance between these components, allowing for fine-grained control over the learning process.

C. Constructing Cross-Domain Representation Mapping

Expanding upon the components of the previously mentioned architecture representation learner, NAS solutions can be explicitly transferred to tasks in other domains through the proposed Bridge. As illustrated in Fig. 1, the following steps focus on constructing a mapping between representation spaces of the source and the target domain, enabling the explicit transfer of neural architecture solutions across domains. By establishing this mapping, the model can effectively leverage the knowledge gained from the source domain to improve the search process and discover high-performing architectures in the target domain.

Establishing a mapping between the representation spaces of source and target domains crucially facilitates knowledge transfer across domains. Let $\mathbb{R}_{\mathcal{S}}$ and $\mathbb{R}_{\mathcal{T}}$ represent the representation spaces of the source ($\mathcal{S}$) and target ($\mathcal{T}$) domains, respectively. The goal is to learn a mapping function $\mathcal{M}: \mathbb{R}_{\mathcal{S}} \to \mathbb{R}_{\mathcal{T}}$ that minimizes the discrepancy between the mapped representations from the source domain and their corresponding counterparts in the target domain. Formally, the objective function to be minimized can be expressed as:

$$\mathcal{L}_{\text{map}}(\mathcal{M}) = \sum_{i=1}^{N} \ell\left(\mathcal{M}(\boldsymbol{z}_s^{(i)}), \boldsymbol{z}_t^{(i)}\right) \tag{16}$$

where $\boldsymbol{z}_s^{(i)} \in \mathbb{R}_{\mathcal{S}}$ and $\boldsymbol{z}_t^{(i)} \in \mathbb{R}_{\mathcal{T}}$ are the i-th sampled representations from the source and target domains, respectively, and sort according to the predicted values; N is the total number of sampled representation pairs; and ℓ is a loss function quantifying the dissimilarity between the mapped source representation and the target representation. The selection of the mapping function $\mathcal{M}$ may vary based on the characteristics of the representation spaces and the desired flexibility and complexity of the mapping. In this work, we exemplify the use of a linear mapping, $\mathcal{M}(\boldsymbol{z}_s) = \boldsymbol{z}_s W + \boldsymbol{b}$, where W and $\boldsymbol{b}$ are learnable parameters representing the weight matrix and bias vector, respectively. Utilizing the *Mean Squared Error* (MSE) loss as ℓ, the mapping effectively learns to project representations from the source domain onto the target domain by optimizing the objective function:

$$\min_{W,\mathbf{b}} \mathcal{L}_{\text{map}}(\mathcal{M}) = \frac{1}{N}\sum_{i=1}^{N}\left\|\boldsymbol{z}_t^{(i)} - \left(\boldsymbol{z}_s^{(i)}W + \boldsymbol{b}\right)\right\|^2. \tag{17}$$

Knowledge transfer primarily impacts population initialization. The transfer mechanism involves encoding source solutions ($O(S^2 \cdot d + S \cdot d^2)$ complexity), linear mapping, and decoding to target solutions (same complexity). This process is efficient, adding minimal overhead (< 0.9% total time, ~190ms per batch on RTX2080Ti).

D. Evolutionary TNAS with Cross-Domain Sequential Transfer

Building on our cross-domain neural architecture mapping, we further design an ESTO algorithm — a population-based approach capable of efficiently adapting the explicit transfer of solutions between domains. Our transfer method and solver are decoupled by design, allowing for the use of various

solvers. However, we chose an evolutionary solver due to its distinct advantages [45]–[47]. Firstly, evolutionary algorithms naturally maintain a diverse population of solutions, enabling the integration of transferred architectures while preserving exploration capabilities. This population-based approach offers more robust knowledge transfer than single-solution methods. Additionally, evolutionary crossover operations provide an effective mechanism for combining transferred knowledge with novel architectural patterns. This synergy allows the search process to leverage prior experience while uncovering new solutions specifically tailored to the target domain. Furthermore, evolutionary methods impose fewer constraints on the architecture representation, making them especially suitable for cross-domain transfer where source and target domains may differ in their search spaces.

This approach leverages optimal neural architecture populations from the source domain to initialize and guide the evolutionary search process in the target domain, efficiently exploring promising regions in the target search space. In the source domain, the framework first identifies the optimal neural architecture population using the pre-trained performance predictor. Top-performing individuals are selected to form an elite population, embodying the accumulated knowledge from the source domain. The overview of the proposed ESTO algorithm in BRIDGE is presented in Algorithm 2. The process unfolds as follows:

1) **Knowledge Encoding (line 1–2):** The selected architectures from the optimal source population are encoded into representation vectors using the source domain's encoder. These encoded representations capture the essence of high-performing architectures discovered in the source domain.
2) **Cross-Domain Mapping (line 3):** The encoded representations are mapped from the source representation space to the target representation space using the learned mapping function $\mathcal{M}$.
3) **Initial Population Seeding (line 4–6):** The mapped representations are then decoded as the initial population for the evolutionary search process, provides a strategic starting point for the NAS to explore the target architecture search space.
4) **Evolutionary Search (line 7–12):** The ESTO process then proceeds with a normal EA, where the optimization is guided by the knowledge transferred from the source domain. The performance predictor, $\mathrm{Rkr}_{\mathcal{T}}(\cdot)$, trained on the target domain, is employed to assess the fitness of new architectures.

A critical consideration in transfer learning involves addressing the challenge of negative transfer —- the fundamental mismatch between transferred knowledge and target task requirements. The proposed BRIDGE integrates multiple structural mechanisms to mitigate this phenomenon. First, our evolutionary optimization strategy preserves population diversity throughout the search process through three key design features: 1) The population-based paradigm maintains exploration capacity even when partial transferred solutions underperform, enabling recovery through subsequent evolutionary operations; 2) A transfer probability parameter regulates the proportion of transferred knowledge in initial populations, ensuring balanced exploitation of prior knowledge while preserving novel solution discovery in the target domain; 3) An adaptive selection mechanism rigorously evaluates transferred architectures against target task objectives, systematically eliminating suboptimal candidates during the evolutionary cycle. By integrating EA with cross-domain transfer capabilities, BRIDGE effectively leverages existing NAS solutions and knowledge from diverse search spaces, potentially leading to more efficient and effective neural architecture search in new domains.

Algorithm 2: The proposed ESTO for BRIDGE

Input: P_{src}, the set of solutions to be transferred obtained by a historical NAS process on the source domain; G, the maximum generation count.
Output: P_{rst}, the final population searched on the target domain.

/* preparations */
1 $\left[\mathrm{Enc}_{\mathcal{S}}(\cdot), \mathrm{Dec}_{\mathcal{S}}(\cdot), \mathrm{Rkr}_{\mathcal{S}}(\cdot)\right] \leftarrow$ learning representations on source domain;
2 $\left[\mathrm{Enc}_{\mathcal{T}}(\cdot), \mathrm{Dec}_{\mathcal{T}}(\cdot), \mathrm{Rkr}_{\mathcal{T}}(\cdot)\right] \leftarrow$ learning representations on target domain;
3 $\mathcal{M} \leftarrow$ learning mapping from $\mathbb{R}_{\mathcal{S}}$ to $\mathbb{R}_{\mathcal{T}}$;
/* sequential transfer */
4 $R_{\mathrm{src}} \leftarrow \mathrm{Enc}_{\mathcal{S}}(P_{\mathrm{src}})$;
5 $R_{\mathrm{tgt}} \leftarrow \mathcal{M}(R_{\mathrm{src}})$;
6 $P_0 \leftarrow \mathrm{Dec}_{\mathcal{T}}(R_{\mathrm{tgt}})$;
/* evolutionary NAS */
7 **for** *g from* 1 *to* G **do**
8 $\quad P_{\mathrm{new}} \leftarrow$ Apply evolutionary operations on P_{g-1};
9 $\quad F_{\mathrm{new}} \leftarrow \mathrm{Rkr}_{\mathcal{T}}(P_{\mathrm{new}})$;
10 $\quad P_g \leftarrow$ Binary tournament selection from $P_g \cup P_{\mathrm{new}}$;
11 **end for**
12 $P_{\mathrm{rst}} \leftarrow P_g$;
13 **return** P_{rst}, *the final population on target domain.*

IV. EXPERIMENTS

This section presents a comprehensive experimental evaluation designed to validate the efficacy of our proposed cross-domain TNAS framework, BRIDGE. We conduct a series of experiments across search spaces with different scales, providing in-depth analysis of the results. Furthermore, to rigorously examine the individual components and design choices of our framework, we devise a set of studies for deeper analysis. These studies serve to elucidate the contribution of each key element of our approach and to provide empirical support for our methodological decisions.[1]

A. Search Spaces and Datasets

To rigorously evaluate BRIDGE, we selected three neural architecture search spaces: NAS-Bench-201, NAS-Bench-101, and DARTS. These spaces progressively increase in

[1]The source code, datasets, and complete experimental configurations will be made publicly available in a dedicated repository at https://github.com/HowBoring/BRIDGE-NAS.

complexity, providing a structured framework for validation. The experimental design implements a graduated complexity paradigm, beginning with more constrained search spaces and progressively advancing to more complex domains. This hierarchical approach serves dual purposes: validating the scalability of our proposed methodology while facilitating systematic investigation of knowledge transfer capabilities in increasingly realistic architectural scenarios. The progression culminates in the evaluation of search spaces of sufficient complexity to address real-world computational challenges.

NAS-Bench-201 Space. [44] This space represents architectures as directed acyclic graphs (DAGs) using an operator-on-edge (OOE) scheme. Each edge in the DAG signifies an operation such as `NONE`, `SKIP-CONNECT`, `CONV-1X1`, `CONV-3X3`, or `AVG-POOL-3X3`. The space comprises 15,625 unique architectures, making it an effective starting point for evaluating NAS methodologies.

NAS-Bench-101 Space. [43] Increasing in complexity, NAS-Bench-101 employs a cell-based DAG architecture using an operator-on-node (OON) scheme. Nodes correspond to operations such as `CONV-3X3`, `CONV-1X1`, and `MAX-POOL-3X3`, with tensor edges connecting them. The search space includes architectures with up to 7 operators and 9 connections, totaling 423,624 unique configurations. This space challenges NAS algorithms by requiring adaptability and scalability.

DARTS Space. [7] Representing much more complexity, DARTS models architectures as computational cells, also structured as DAGs but with a more diverse set of operations. Each cell consists of 7 nodes, where input tensors flow through an ordered structure. Supported operations include `MAX-POOL-3X3`, `AVG-POOL-3X3`, `SKIP-CONNECT`, `SEP-CONV-3X3`, `SEP-CONV-5X5`, `DIL-CONV-3X3`, and `DIL-CONV-5X5`. DARTS employs normal cells to maintain spatial resolution and reduction cells for downsampling. With over 10^{18} potential architecture combinations, this space serves as a rigorous benchmark for evaluating large-scale NAS performance.

B. Building Representations

As described in Section III-B, we established distinct representations for each architectural search space. Table I details the hyperparameter configurations for training representation learners. All experiments were conducted on a single NVIDIA RTX2080Ti GPU.

Optimization employed AdamW with a learning rate of 2×10^{-5} and weight decay of 1×10^{-6}, with modulation following Vaswani et al. [39] and a 10% warm-up phase. For evaluation, we used established benchmarks for NAS-Bench-101 and NAS-Bench-201. For DARTS, following CTNAS [34], we conducted supernet training (200 epochs), sampled 1000 architectures, and obtained validation accuracies through weight inheritance.

Table II presents performance metrics of our representation learner. The upper section shows reconstruction accuracy, with our method achieving nearly 100% across all search spaces. Since VAE parameters remain unfrozen to integrate performance information, a slight reduction in reconstruction accuracy may occur.

TABLE I
HYPER-PARAMETER CONFIGURATIONS FOR NEURAL ARCHITECTURE REPRESENTATION LEARNING ON DIFFERENT SEARCH SPACES

		NB201	NB101	DARTS
Training Settings	*pt-size* [a]	5,000 (50%)	12,000 (30%)	30,000 (–) [b]
	ft-size	1,000 (10%)	4,000 (1%)	1,500 (–)
	pt-epoch	200	200	200
	ft-epoch	200	200	200
Model Settings	*repr-dim*	64	64	128
	model-dim	128	128	512
	ff-dim	256	256	2,048
	n-enc-lyr	4	4	4
	n-dec-lyr	4	4	4
	n-head	4	4	4

pt = pre-training, *ft* = fine-tuning, *repr* = representation, *ff* = feed-forward, *enc-lyr* = encoder layer, *dec-lyr* = decoder layer.
[a] The number of samples used for training, represented in parentheses as their proportion to the entire search space. Same for other sizes.
[b] The DARTS search space encompasses a vast number of neural architectures, making it impractical to determine a specific proportion.

TABLE II
COMPARATIVE PERFORMANCE ANALYSIS OF NEURAL ARCHITECTURE REPRESENTATION LEARNING METHODS

			NB201	NB101	DARTS
Reconstruction Accuray (%)	↑	*arch2vec* [42]	99.99	98.84	–
		SVGe [38]	99.99	99.57	99.63
		DGMG [48]	99.97	98.99	99.29
		BRIDGE (Ptd.)	99.99	99.90	99.25
		BRIDGE (Ftd.)	99.06	99.99	99.12
Predicdtion Kendall's τ-b	↑	NAO [25]	0.526	0.775	–
		TNASP [49]	0.724	0.820	–
		ReNAS-6 [31]	–	0.816	–
		BRIDGE	**0.827**	**0.848**	0.811 [a]

[a] The result is based on the labels we collected on the DARTS space.

The lower section presents ranker fine-tuning results using *Kendall's τ-b Correlation Coefficient* (K-Tau). Our approach achieves coefficients of 0.848 (NAS-Bench-101) and 0.827 (NAS-Bench-201), surpassing NAO [29], TNASP [52], and ReNAS [31]. In DARTS, we achieve 0.811, supporting effective architecture transfer. These results demonstrate our ranking predictor's superior alignment with actual performance rankings.

Importantly, our primary objective is not architecture performance prediction. Rather, these results demonstrate our method's effectiveness in extracting intrinsic architectural features, which underpins the fundamental theory of our proposed cross-domain transfer learning method for neural architectures.

C. Transfer from Simple to Complex Domains

We conducted preliminary validation of the proposed method through experiments involving transfer learning from simple to complex domains, specifically using solutions from the NAS-Bench-201 space to enhance the search in the NAS-Bench-101

TABLE III
RESULTS OF NEURAL ARCHITECTURE SEARCH ON NAS-BENCH-101 SPACE

Profile	Optimal Acc. (%)	Average Acc. (%)	#Evaluation	Cost (sec)	P-Value [a]	Method Type
ENAS [6]	92.54	91.83 ± 0.42	–	–	6.52E-08	Reinforcement Learning
FBNet [50]	93.98	92.29 ± 1.25	–	–	2.01E-03	Gradient-based
CTNAS [34]	94.14	93.93 ± 0.12	–	–	2.08E-01	Predictor
ReNAS [31]	94.07	93.95 ± 0.09	–	–	3.08E-01	Predictor
SVGe [38]	93.88	–	–	–	–	Gaussian Process
Naïve GA (baseline)	93.85	93.71 ± 0.11	500	26	7.19E-06	Evolutionary
BRIDGE (ours)	**94.17**	**93.99**±0.08	80	**8**		Evolutionary + Transfer
PSO [51]	93.77	93.71 ± 0.04	1,000	40	7.18E-10	Evolutionary
PSO + BRIDGE	93.96	93.92 ± 0.04	80	15		Evolutionary + Transfer
MSNAS [8]	94.07	93.93 ± 0.10	1,400	63	3.17E-03	Evolutionary
MSNAS + BRIDGE	**94.32**	**94.13**±0.15	100	16		Evolutionary + Transfer

[a] P-values in the first part are calculated relative to BRIDGE, while those in subsequent parts are relative to their respective baselines. A p-value below 0.05 indicates that our method outperforms the comparison algorithm or the non-transfer baseline significantly.

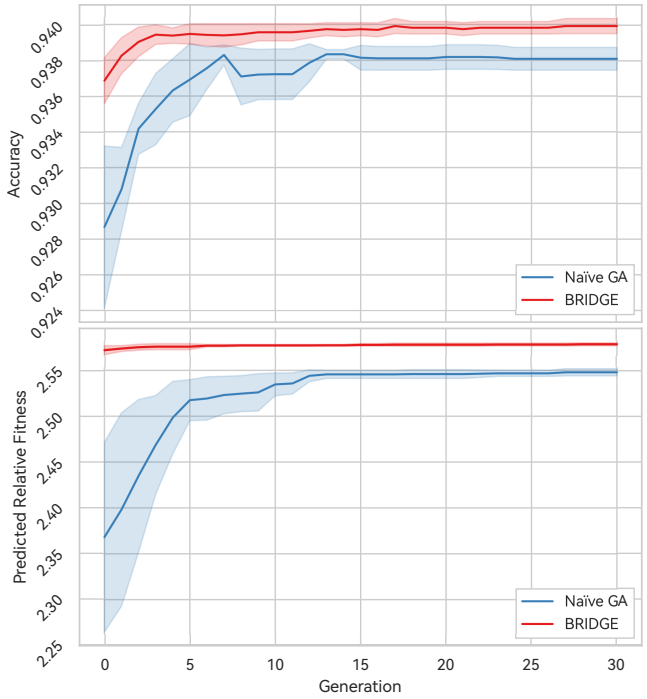


Fig. 5. Comparative analysis of evolutionary neural architecture search performance on NAS-Bench-101 with and without BRIDGE's cross-domain ESTO mechanism. *Top:* The ground truth accuracy of explored architectures over generations. *Bottom:* Predicted relative fitness scores from the ranker module, which guides the NAS process as the evaluation strategy.

space. The mapping matrix $\mathcal{M}$ is constructed by sampling 1,000 solutions from the training sets of both sides separately. We selected Genetic Algorithm (GA) [53], a widely used method in NAS, as the evolutionary search solver. Following the settings in [10], the GA solver used a population size of 20, with *Single Point Crossover* and *Polynomial Mutation*, both set at a probability of 0.5, and *Binary Tournament Selection*.

The results, presented in Table III, compare our method against a baseline established by the same GA solver but excluding the proposed cross-domain transfer method (Naïve GA). Our method achieves significant performance gains, with an optimal accuracy of 94.17% and an average accuracy of 93.99%, substantially outperforming the baseline metrics. Statistical analysis across 10 independent trials confirms the significance of these improvements. To further illustrate the impact of transfer learning on the search process, Fig. 5 shows the generation-by-generation trend of the population's optimal solution. In the early stages, the transferred solutions create a high-quality and diverse initial population, thus enabling the exploration of more optimal solution regions from the onset. In summary, our proposed transfer learning method effectively accelerates the process of neural architecture search, allowing it to converge earlier to a better result region.

Moreover, our transfer mechanism is explicitly designed to be solver-agnostic within the evolutionary computing framework, emphasizing broad applicability across different optimization approaches. The modular architecture of our framework facilitates straightforward integration with existing NAS methodologies, creating potential for enhanced architectural optimization across diverse search strategies. To rigorously validate this versatility, we conducted comprehensive experiments detailed in Table III, demonstrating successful integration with two advanced evolutionary approaches: *particle swarm optimization* (PSO) from EPCNAS [51] and *memetic algorithm* (MA) from MSNAS [8]. The experimental results show notable improvements, with PSO and MA achieving performance gains of 0.19% and 0.25% respectively over their baseline implementations. Importantly, these improvements were achieved while simultaneously reducing the computational overhead, requiring fewer evaluations and iterations to converge. These findings provide strong empirical evidence for the framework's solver-agnostic nature and its capacity to enhance various evolutionary optimization strategies.

D. Transfer to More Challenging Domains

To evaluate the generalizability of our transfer learning framework, we conducted extensive experiments in the DARTS architectural space, which presents substantially increased complexity in the search domain. The DARTS space exhibits significant architectural heterogeneity compared to NAS-Bench-101 and NAS-Bench-201, characterized by an expanded operational vocabulary, increased operational density per cell, and

a dual-cell type configuration. These architectural disparities present substantial challenges for knowledge transfer, providing a rigorous validation framework for our methodology.

Our experimental protocol focused on knowledge transfer from the NAS-Bench-101 domain to the DARTS search space. We maintained consistent hyperparameter configurations with those detailed in Section IV-C. The experimental procedure consisted of predictor-guided convergence search followed by comprehensive training of the top 10 architectures, selected based on predictive performance metrics on CIFAR-10. Following established methodological protocols [5]–[8], [23], we implemented data augmentation and *cutout* enhancement techniques during the complete training phase.

Table IV summarizes the performance of our approach in comparison to baseline methods. Methods like DARTS (97.24% at 4 GPU days) and MSNAS (97.32% at 0.23 GPU days), demonstrate a clear trend: efficient search mechanisms can substantially reduce resource expenditure. When transfer learning is incorporated (as in EMT-NAS and MTNAS, both leveraging EA + TL), the models benefit from additional performance gains and cost reductions.

Our proposed method, BRIDGE, which also adopts an EA + TL strategy, attains the highest accuracy (97.33% ± 0.06) with an impressively low search cost of 0.21 GPU days. This outcome not only outperforms the naïve GA baseline (96.62% at 0.3 GPU days) but also establishes a new benchmark by effectively transferring high-quality solutions from simpler domains to more complex ones within the DARTS search space.

These results underscore the adaptability of our method to varied and complex architectural designs. This adaptability is crucial for addressing the challenges posed by large-scale search spaces, thus validating the practical applicability of our framework in real-world scenarios. Moreover, the successful transfer across domains with high dissimilarity suggests that our representation learning approach effectively extracts the underlying semantic information of neural architectures. This capability to bridge disparate domains could significantly advance the field of automated machine learning, enabling more efficient and effective architecture discovery across a wide range of tasks and domains.

TABLE IV
COMPARISONS WITH STATE-OF-THE-ART NAS MODELS ON CIFAR-10 USING DARTS SEARCH SPACE

Profile	Average Acc. (%)	Search Cost (GPU Days)	Method Type
ResNet-110 [1]	93.40	–	Manual
DenceNet-BC [54]	96.54	–	Manual
AmoebaNet-A [23]	96.66 ± 0.06	3150	EA
ENAS [6]	97.11	0.5	RL
NAO [25]	97.02	200	GD
DARTS [7]	97.24 ± 0.09	4	GD
MSNAS [8]	97.32 ± 0.08	0.23	EA
EMT-NAS [12]	97.04 ± 0.04	0.42	EA + TL
MTNAS [13]	97.25 ± 0.04	0.25	EA + TL
Naïve GA (baseline)	96.62 ± 0.15	0.3	EA
BRIDGE (ours)	**97.33** ± 0.06	**0.21**	EA + TL

EA = Evolutionary Algorithm, RL = Reinforcement Learning, GD = Gradient-based, TL = Transfer Learning.

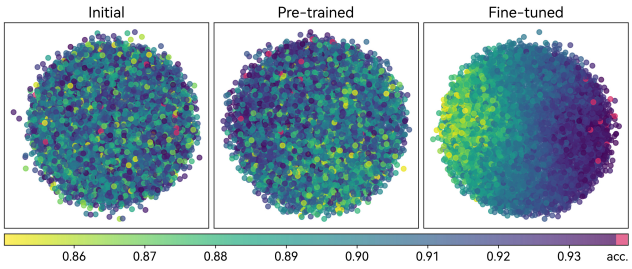


Fig. 6. Visualization of the learned representation space for NAS-Bench-101 architectures across different training stages: initial (left), after pre-training (middle), and after fine-tuning (right). Colors indicate architecture performance (accuracy), with warmer colors representing higher accuracy.

V. DEEPER ANALYSIS

A. Latent Representation Space Observation

To provide a more intuitive representation of the evolving distribution within the latent representation space throughout the training process, we employ *Multi-Dimensional Scaling* (MDS) [55] for visualization. Additionally, we color-code the sampled points according to their ground-truth labels to facilitate interpretation. As illustrated in Fig. 6, neural architectures with similar performance metrics exhibit spatial proximity within the latent representation space across various stages of the representation learning process. This aggregation demonstrates a gradual, stepwise increase in coherence over training process.

The observed clustering behavior aligns with the inherent properties of VAEs. During the pre-training phase, the VAE naturally tends to position similar neural architectures in close proximity within the latent space. Moreover, architectures with structural similarities have a higher probability of exhibiting comparable performance characteristics. This intrinsic organization provides a robust foundation for the subsequent training of the performance ranker. In the fine-tuning phase, the introduction of supervisory signals derived from ground-truth labels further enhances the model's capacity to capture features that contribute to performance differentials. This refinement is reflected in the latent space distribution, where representations of neural architectures with similar performance metrics display remarkable aggregation. Furthermore, mapping the evolutionary dynamics onto the latent space (Fig. 7) illustrates how BRIDGE accelerates evolution. Initializing with transferred high-quality, diverse solutions enhances convergence speed and discovery of top-performing architectures.

B. The Impact of The Amount of Training Data Used

As detailed in Section III-B, our representation learning framework employs a two-phase training strategy: unsupervised pre-training followed by supervised fine-tuning. This approach enables comprehensive capture of architectural characteristics while maintaining transferability across domains. Since representation quality, dependent on data volume, is crucial for transfer effectiveness, we studied the impact of dataset size on performance. This assesses our approach's scalability and robustness under varying data availability.

This article has been accepted for publication in IEEE Transactions on Evolutionary Computation. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2025.3565326

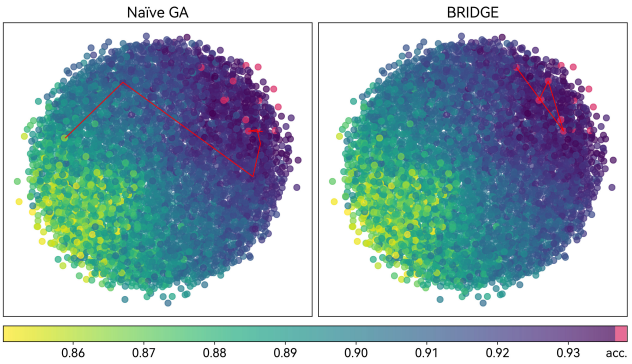


Fig. 7. Evolutionary dynamics with and without Transfer Learning on NAS-Bench-101. The figure compares the optimization trajectory on NAS-Bench-101 with (right panel) and without (left panel) transfer learning. The trajectory is visualized by tracking the optimal solution every ten generations.

TABLE V
NEURAL ARCHITECTURE REPRESENTATION LEARNING PERFORMANCE DURING THE PRETRAIN STAGE ON NAS-BENCH-101, EVALUATED ACROSS DIFFERENT DATASET SIZES

Unlabeled Data	**1%**	**5%**	**10%**	**30%**	**50%**
VAE Acc. (%)	78.82	89.46	99.91	99.56	99.97
Cost (GPU Days)	0.08	0.08	0.08	0.11	0.15

Table V presents critical performance metrics from the pre-training phase across varying fractions of the NAS-Bench-101 dataset. We specifically analyze two key indicators: the VAE reconstruction accuracy and computational overhead (measured in GPU days). Pre-training results show VAE accuracy strongly correlates with data volume up to 10% (achieving 99.91% accuracy at 0.08 GPU days), with diminishing returns thereafter (e.g., 30% data yields 99.56% accuracy at higher cost). This demonstrates efficient representation learning even with limited unlabeled data, supporting practical application where data is scarce or costly.

During fine-tuning, as shown in Table VI, while VAE reconstruction accuracy remains high (>99.90%) across dataset sizes, predictor performance (K-Tau) is more sensitive, improving with more labeled data (peaking at 0.865 K-Tau with 5% data). This sensitivity also affects the learned representation space's structure. This sensitivity manifests not only in prediction accuracy but also influences the distribution characteristics of the learned representation space. Larger labeled datasets yield a more structured representation space, enabling better inter-domain mapping and enhancing transfer performance (e.g., using 5% data yields top results: 97.36% source, 94.32% target accuracy), highlighting the benefit for cross-domain transferability.

In summary, while the VAE demonstrates resilience to limited data, the performance predictor and transfer learning stages benefit significantly from larger datasets. This highlights the importance of data acquisition and curation in transfer learning scenarios, particularly when aiming to optimize performance in target domains with limited labeled data.

TABLE VI
NEURAL ARCHITECTURE REPRESENTATION LEARNING PERFORMANCE DURING THE FINETUNING STAGE ON NAS-BENCH-101, EVALUATED ACROSS DIFFERENT DATASET SIZES

Labeled Data	**0.1%**	**1%**	**5%**
Kendall's τ-b	0.738	0.848	0.865
Cost (GPU Days)	0.01	0.02	0.02
Optimal Result [a]			
— As Source (%)	96.56	97.33	97.36
— As Target (%)	94.02	94.17	94.32

[a] "As Source" and "As Target" refer to transferring to DARTS and from NB201, respectively.

TABLE VII
PERFORMANCE COMPARISON OF DIFFERENT MAPPING METHODS FOR TRANSFER LEARNING FROM NAS-BENCH-201 TO NAS-BENCH-101

Method	**Optimal Acc. (%)**	**Average Acc. (%)**	**P-Value** [a]
No Transfer	93.85	93.77 ± 0.11	7.19E-06
Linear Mapping	94.17	93.99 ± 0.08	–
Non-linear Mapping	94.09	93.97 ± 0.06	4.62E-01

[a] All P-values are calculated relative to the Linear Mapping method based on 10 times independent experiments.

C. Explore Different Mapping Methods

As presented in Section III-C, we adopted a linear mapping approach for its computational efficiency, transparency, and simplicity in validating explicit solution transfer between heterogeneous domains. Our empirical analysis reveals that the learned representations exhibit smooth, well-structured distributions in the latent space, naturally facilitating inter-domain transfer through linear transformations. Linear mapping efficacy is supported empirically by the smooth, structured latent space distributions observed, and theoretically by our framework design which preserves architectural similarities in a continuous space suitable for linear transformation.

To rigorously evaluate the impact of mapping complexity on transfer performance, we conducted a comparative study between linear and nonlinear mapping approaches. As shown in Table VII, both approaches achieved comparable performance metrics, with no statistically significant difference observed ($p = 0.462 > 0.05$). This suggests representation quality, not mapping complexity, primarily determines transfer success.

D. Hyper-parameters Sensitivity Analysis

Key Transformer hyperparameters (e.g. attention heads, embedding dimension, layer depth) influence representation learning. Our analysis shows their impact diminishes as model scale increases. A grid search over Transformer depth (2, 3, 4), dimension (64, 128), and heads (2, 4, 8) showed low VAE accuracy standard deviation (10^{-2} overall, dropping to 3×10^{-3} for depth ≥ 3), indicating robustness, especially for deeper models. A parallel coordinates plot that visualizes the hyper-parameter configurations is provided to further illustrate these findings in Fig. 8.

Furthermore, hyper-parameter selection within the evolutionary search component of BRIDGEwas systematically investigated.

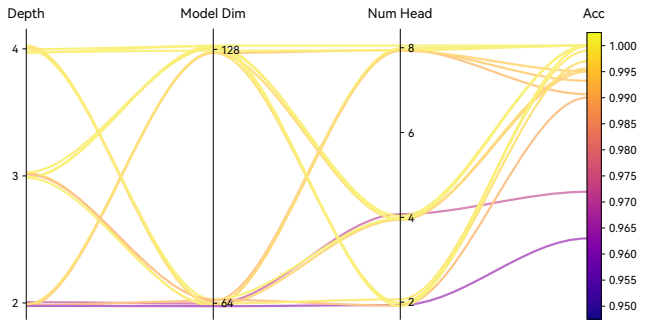


Fig. 8. Parallel coordinates plot showing the accuracy of the proposed representation learner on NAS-Bench-101 under different hyper-parameter settings. Each line corresponds to a unique combination of network depth, model dimension, and number of attention heads, while the color scale indicates the resulting accuracy (darker lines indicate lower accuracy and brighter lines indicate higher accuracy).

As shown in Table VIII, we explored a hyper-parameter space defined by population size (10, 20, 30), crossover probability (0.1–0.9), and mutation probability (0.1–0.9). We assessed their influence using Pearson Correlation Coefficient and random forest feature importance, providing insights into linear and non-linear relationships.

Population Size. Correlation increased from 0.460 (GA) to 0.629 (TL), and importance significantly rose from 0.287 (GA) to 0.754 (TL). This highlights population size as more crucial in TL, likely due to enhanced exploration and adaptation to new search spaces, promoting diversity and search efficacy.

Crossover Probability. Correlation decreased from 0.374 (GA) to −0.039 (TL), and importance dropped from 0.447 (GA) to 0.051 (TL). This indicates crossover is less influential in TL. While beneficial in GA for exploring solution spaces, in TL, transferred knowledge reduces crossover's necessity as initial populations are already diverse and high-quality.

Mutation Probability. Correlation increased from 0.095 (GA) to 0.527 (TL), suggesting higher mutation is more beneficial in TL for exploring new domains, enhancing diversity and preventing convergence. However, importance decreased from 0.267 (GA) to 0.195 (TL). Despite increased correlation, mutation's relative contribution is less vital in TL than GA because knowledge transfer provides a better starting point, reducing reliance on mutation for initial discovery.

In conclusion, population size and mutation probability are more critical in TL than GA, requiring larger, more diverse populations for new tasks. Reduced crossover impact in TL suggests effective knowledge transfer provides a strong starting point, lessening the need for crossover-driven exploration. This supports BRIDGE's capacity for high-quality, diverse knowledge transfer in heterogeneous domains.

VI. CONCLUSIONS

This paper introduces BRIDGE, a framework that enhances evolutionary TNAS through systematic cross-domain knowledge transfer. The framework facilitates efficient transfer of high-performing architectural solutions across diverse search spaces while minimizing computational costs via learned latent representations. Extensive empirical evaluations show that BRIDGE outperforms state-of-the-art NAS methods, significantly improving both optimal and average accuracy across NAS-Bench-201, NAS-Bench-101, and DARTS search spaces. The framework's effectiveness is further confirmed through manifold visualization and detailed performance distribution analysis.

While BRIDGE offers a promising approach for cross-domain knowledge transfer in NAS, several challenges remain. A key limitation is the feasibility of representation learning, particularly the scarcity of labeled data in complex, heterogeneous search spaces and the added complexity from hyperparameter choices. Future research should explore self-supervised techniques for few-shot and zero-shot learning and enhance unsupervised methods to address hyperparameter differences. Another challenge is mitigating negative transfer between dissimilar domains. Future work should develop adaptive transfer mechanisms based on domain similarity, establish criteria for source domain selection, and create systems for early negative transfer detection. Additionally, the exponential growth of architecture search spaces affects exploration efficiency and knowledge transfer effectiveness. Future studies should focus on hierarchical search space decomposition and multi-objective optimization, improving BRIDGE's scalability for real-world, resource-constrained applications. Overall, BRIDGE provides a solid foundation for future advancements in cross-domain architectural optimization and more efficient NAS methodologies.

TABLE VIII
ANALYSIS OF EVOLUTIONARY HYPER-PARAMETERS FOR TRANSFER LEARNING FROM NAS-BENCH-201 TO NAS-BENCH-101

Param. Range	**Pop. Size** 10, 20 and 30		**Cros. Prob.** 0.1 to 0.9		**Mut. Prob.** 0.1 to 0.9	
	GA	**TL**	**GA**	**TL**	**GA**	**TL**
Correlation	0.460	0.629	0.374	0.039	0.095	0.527
Importance	0.287	0.754	0.447	0.051	0.267	0.195

GA signifies the baseline, Naïve GA, whereas **TL** denotes the proposed BRIDGE.

ACKNOWLEDGMENTS

This work was supported in part by the National Key R&D Program of China (Grant No. 2022YFC3801700), in part by the National Natural Science Foundation of China (Grant Nos. 62402069 and 62402410), in part by the Postdoctoral Fellowship Program of the CPSF (Grant No. GZB20240915), in part by the Natural Science Foundation of Chongqing (Grant No. CSTB2022NSCQ-MSX1285), by the Guangdong Provincial Project (Grant No. 2023QN10X025), by the Guangdong Basic and Applied Basic Research Foundation (Grant No. 2023A1515110131), and by the Guangzhou Municipal Science and Technology Bureau (Grant No. 2024A04J4454).

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2016, pp. 770–778.

[2] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2016, pp. 779–788.

[3] M. X. Chen, O. Firat, A. Bapna, M. Johnson, W. Macherey, G. Foster, L. Jones, M. Schuster, N. Shazeer, N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, Z. Chen, Y. Wu, and M. Hughes, "The best of both worlds: Combining recent advances in neural machine translation," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. Melbourne, Australia: Association for Computational Linguistics, 2018, pp. 76–86.

[4] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *International Conference on Learning Representations*, 2017.

[5] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2018, pp. 8697–8710.

[6] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," in *Proceedings of the International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. G. Dy and A. Krause, Eds., vol. 80. PMLR, 2018, pp. 4092–4101.

[7] H. Liu, K. Simonyan, and Y. Yang, "DARTS: differentiable architecture search," in *International Conference on Learning Representations*, 2019.

[8] J. Dong, B. Hou, L. Feng, H. Tang, K. C. Tan, and Y. Ong, "A cell-based fast memetic algorithm for automated convolutional neural architecture design," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 11, pp. 9040–9053, 2023.

[9] B. Baker, O. Gupta, R. Raskar, and N. Naik, "Accelerating neural architecture search using performance prediction," in *International Conference on Learning Representations*, 2018.

[10] B. Hou, J. Dong, L. Feng, and M. Qiu, "Efficient two-stage evolutionary search of convolutional neural architectures based on cell independence analysis," in *Neural Information Processing, ICONIP 2021*, ser. Communications in Computer and Information Science, vol. 1516. Springer, 2021, pp. 599–607.

[11] Z. Lu, G. Sreekumar, E. D. Goodman, W. Banzhaf, K. Deb, and V. N. Boddeti, "Neural architecture transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 9, pp. 2971–2989, 2021.

[12] P. Liao, Y. Jin, and W. Du, "EMT-NAS: transferring architectural knowledge between tasks from different datasets," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2023, pp. 3643–3653.

[13] X. Zhou, Z. Wang, L. Feng, S. Liu, K. Wong, and K. C. Tan, "Toward evolutionary multitask convolutional neural architecture search," *IEEE Transactions on Evolutionary Computation*, vol. 28, no. 3, pp. 682–695, 2024.

[14] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Completely automated CNN architecture design based on blocks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 4, pp. 1242–1254, 2020.

[15] Y. Liu, Y. Tang, Z. Lv, Y. Wang, and Y. Sun, "Bridge the gap between architecture spaces via A cross-domain predictor," in *Advances in Neural Information Processing Systems*, vol. 35, 2022.

[16] K. C. Tan, L. Feng, and M. Jiang, "Evolutionary transfer optimization - A new frontier in evolutionary computation research," *IEEE Computational Intelligence Magazine*, vol. 16, no. 1, pp. 22–33, 2021.

[17] X. Xue, C. Yang, Y. Hu, K. Zhang, Y. Cheung, L. Song, and K. C. Tan, "Evolutionary sequential transfer optimization for objective-heterogeneous problems," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 6, pp. 1424–1438, 2022.

[18] X. Xue, C. Yang, L. Feng, K. Zhang, L. Song, and K. C. Tan, "Solution transfer in evolutionary optimization: An empirical study on sequential transfer," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2023.

[19] Y. Gao, H. Yang, P. Zhang, C. Zhou, and Y. Hu, "Graphnas: Graph neural architecture search with reinforcement learning," *ArXiv*, vol. abs/1904.09981, 2019.

[20] B. Lyu, S. Wen, K. Shi, and T. Huang, "Multiobjective reinforcement learning-based neural architecture search for efficient portrait parsing," *IEEE Transactions on Cybernetics*, vol. 53, pp. 1158–1169, 2021.

[21] C.-H. Hsu, S.-H. Chang, D.-C. Juan, J.-Y. Pan, Y.-T. Chen, W. Wei, and S.-C. Chang, "Monas: Multi-objective neural architecture search using reinforcement learning," *ArXiv*, vol. abs/1806.10332, 2018.

[22] Y. Liu, Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "A survey on evolutionary neural architecture search," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, pp. 550–570, 2020.

[23] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, 2019, pp. 4780–4789.

[24] X. Zhou, A. K. Qin, M. Gong, and K. C. Tan, "A survey on evolutionary construction of deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 5, pp. 894–912, 2021.

[25] R. Luo, F. Tian, T. Qin, E. Chen, and T. Liu, "Neural architecture optimization," in *Advances in Neural Information Processing Systems*, 2018, pp. 7827–7838.

[26] J. Huang, B. Xue, Y. Sun, M. Zhang, and G. G. Yen, "Split-level evolutionary neural architecture search with elite weight inheritance." *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, 2023.

[27] M. Huang, Z. Huang, C. Li, X. Chen, H. Xu, Z. Li, and X. Liang, "Arch-graph: Acyclic architecture relation predictor for task-transferable neural architecture search," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2022, pp. 11 871–11 881.

[28] F. X. Han, K. G. Mills, F. Chudak, P. Riahi, M. Salameh, J. Zhang, W. Lu, S. Jui, and D. Niu, "A general-purpose transferable predictor for neural architecture search," in *Proceedings of the SIAM International Conference on Data Mining*, S. Shekhar, Z. Zhou, Y. Chiang, and G. Stiglic, Eds. SIAM, 2023, pp. 721–729.

[29] B. Deng, J. Yan, and D. Lin, "Peephole: Predicting network performance before training," *CoRR*, vol. abs/1712.03351, 2017.

[30] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L. Li, L. Fei-Fei, A. L. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Computer Vision - ECCV*, ser. Lecture Notes in Computer Science, vol. 11205. Springer, 2018, pp. 19–35.

[31] Y. Xu, Y. Wang, K. Han, Y. Tang, S. Jui, C. Xu, and C. Xu, "Renas: Relativistic evaluation of neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Computer Vision Foundation / IEEE, 2021, pp. 4411–4420.

[32] C. White, W. Neiswanger, and Y. Savani, "BANANAS: bayesian optimization with neural architectures for neural architecture search," in *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, 2021, pp. 10 293–10 301.

[33] Y. Sun, H. Wang, B. Xue, Y. Jin, G. G. Yen, and M. Zhang, "Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 350–364, 2020.

[34] Y. Chen, Y. Guo, Q. Chen, M. Li, W. Zeng, Y. Wang, and M. Tan, "Contrastive neural architecture search with neural architecture comparators," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Computer Vision Foundation / IEEE, 2021, pp. 9502–9511.

[35] W. Li, S. Gong, and X. Zhu, "Neural graph embedding for neural architecture search," in *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, 2020, pp. 4707–4714.

[36] H. Shi, R. Pi, H. Xu, Z. Li, J. T. Kwok, and T. Zhang, "Bridging the gap between sample-based and one-shot neural architecture search with BONAS," in *Advances in Neural Information Processing Systems*, 2020.

[37] W. Wen, H. Liu, Y. Chen, H. H. Li, G. Bender, and P. Kindermans, "Neural predictor for neural architecture search," in *Computer Vision - ECCV*, ser. Lecture Notes in Computer Science, vol. 12374. Springer, 2020, pp. 660–676.

[38] J. Lukasik, D. Friede, A. Zela, F. Hutter, and M. Keuper, "Smooth variational graph embeddings for efficient neural architecture search," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–8.

[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[40] B. Guo, T. Chen, S. He, H. Liu, L. Xu, P. Ye, and J. Chen, "Generalized global ranking-aware neural architecture ranker for efficient image classifier search," in *Proceedings of the 30th ACM International Conference on Multimedia*. ACM, 2022, pp. 3730–3741.

[41] Y. Yi, H. Zhang, W. Hu, N. Wang, and X. Wang, "Nar-former: Neural architecture representation learning towards holistic attributes prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 7715–7724.

[42] S. Yan, Y. Zheng, W. Ao, X. Zeng, and M. Zhang, "Does unsupervised architecture representation learning help neural architecture search?" in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020.

[43] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "Nas-bench-101: Towards reproducible neural architecture search," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 2019, pp. 7105–7114.

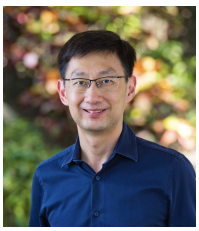
[44] X. Dong, L. Liu, K. Musial, and B. Gabrys, "NATS-Bench: Benchmarking NAS algorithms for architecture topology and size," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3634–3646, 2022.
[45] J. Liang, Y. Zhang, K. Chen, B. Qu, K. Yu, C. Yue, and P. N. Suganthan, "An evolutionary multiobjective method based on dominance and decomposition for feature selection in classification," *Science China Information Sciences*, vol. 67, no. 2, p. 120101, 2024.
[46] P. Yang, L. Zhang, H. Liu, and G. Li, "Reducing idleness in financial cloud services via multi-objective evolutionary reinforcement learning based load balancer," *Science China Information Sciences*, vol. 67, no. 2, p. 120102, 2024.
[47] H. Hao, X. Zhang, and A. Zhou, "Enhancing saeas with unevaluated solutions: a case study of relation model for expensive optimization," *Science China Information Sciences*, vol. 67, no. 2, p. 120103, 2024.
[48] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. W. Battaglia, "Learning deep generative models of graphs," *CoRR*, vol. abs/1803.03324, 2018.
[49] S. Lu, J. Li, J. Tan, S. Yang, and J. Liu, "TNASP: A transformer-based NAS predictor with a self-evolution framework," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021, pp. 15 125–15 137.
[50] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Computer Vision Foundation / IEEE, 2019, pp. 10 734–10 742.
[51] J. Huang, B. Xue, Y. Sun, M. Zhang, and G. G. Yen, "Particle swarm optimization for compact neural architecture search for image classification," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 5, pp. 1298–1312, 2023.
[52] Y. Sun, G. G. Yen, and M. Zhang, *End-to-End Performance Predictors*. Cham: Springer International Publishing, 2023, pp. 237–255.
[53] J. H. Holland, "Genetic algorithms," *Scientific American*, vol. 267, no. 1, pp. 66–73, 1992.
[54] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2017, pp. 2261–2269.
[55] J. Douglas Carroll and P. Arabie, "Chapter 3 - multidimensional scaling," in *Measurement, Judgment and Decision Making*, ser. Handbook of Perception and Cognition (Second Edition), M. H. Birnbaum, Ed. San Diego: Academic Press, 1998, pp. 179–250.

Boyu Hou received the B.Eng. degree from the College of Computer Science, Chongqing University, Chongqing, China, in 2019, where he is currently pursuing the Ph.D. degree. His current research interests include deep neural network as well as transfer learning and optimization.

Liang Feng (Senior Member, IEEE) received the Ph.D. degree from the School of Computer Engineering, Nanyang Technological University, Singapore, in 2014. He is currently a Professor with the College of Computer Science, Chongqing University, Chongqing, China. His research interests mainly include computational and artificial intelligence, machine learning, multi-agent system as well as transfer learning and optimization. He is Associate Editor of the IEEE Transactionss on Evolutionary Computation, IEEE Transactions on Emerging Topics in Computational Intelligence, IEEE Computational Intelligence Magazine, and IEEE Transactions on Cognitive and Developmental Systems. He is also the Founding Chair of the IEEE CIS Intelligent Systems Applications Technical Committee Task Force on Transfer Learning & Transfer Optimization.

Xuefeng Chen (Member, IEEE) received the Ph.D. degree from the School of Computer Science and Engineering, University of New South Wales, Sydney, NSW, Australia, in 2021. He is currently a Lecturer with the College of Computer Science, Chongqing University, Chongqing, China. His research interests include submodular optimization, evolutionary computation, big data mining and analytics, and spatial-temporal database.

Jing Tang (Member, IEEE) received the B.Eng. degree from the University of Science and Technology of China, in 2012, and the Ph.D. degree from the Nanyang Technological University (NTU), in 2018. He is currently an Assistant Professor in the Data Science and Analytics Thrust at The Hong Kong University of Science and Technology (Guangzhou), and an Assistant Professor in the Division of Emerging Interdisciplinary Areas at The Hong Kong University of Science and Technology. Prior to joining HKUST, he was a research assistant professor with the National University of Singapore and a research fellow with NTU. His research focuses on big data management and analytics, social network and graph analysis, machine learning and large language models, and blockchains.



Kay Chen Tan (Fellow, IEEE) received the B.Eng. degree (with First-Class Hons.) and the Ph.D. degree from the University of Glasgow, Glasgow, U.K., in 1994 and 1997, respectively. He is currently the Head and Chair Professor of Computational Intelligence with the Department of Data Science and Artificial Intelligence, The Hong Kong Polytechnic University, Hong Kong SAR. Prof. Tan was the Editor-in-Chief of IEEE Transactions On Evolutionary Computation from 2015 to 2020, and IEEE Computational Intelligence Magazine from 2010 to 2013, and currently serves as an Editorial Board member of 10+ journals. He served as the Vice-President (Publications) of the IEEE Computational Intelligence Society, USA, from 2021 to 2024, and currently serves as the Chief Co-Editor of Springer Book Series on Machine Learning: Foundations, Methodologies, and Applications.

Xiaofeng Liao (Fellow, IEEE) received the B.S. and M.S. degrees in mathematics from Sichuan University, Chengdu, China, in 1986 and 1992, respectively, and the Ph.D. degree in circuits and systems from the University of Electronic Science and Technology of China, Chengdu, in 1997.

He is currently a Professor and the Dean of the College of Computer Science, Chongqing University. He is also a Yangtze River Scholar of the Ministry of Education of China, Beijing, China. He holds five patents, and published four books and over 300 international journal and conference papers. His current research interests include optimization and control, machine learning, privacy protection, neural networks, and bifurcation and chaos.

Prof. Liao serves as an Associate Editor of the IEEE Transactions on Neural Networks and Learning Systems, IEEE Transactions on Cybernetics, Chinese Journal of Electronics, Computer Science, Big Data Mining and Analytics.