Information Processing and Management 63 (2026) 104892

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/ipm

Towards distribution-aware active learning for data-efficient neural architecture predictor

Caiyang Yu [a,b], Yifan Wang [b], Chenwei Tang [b], Wei Ju [b], Xianggen Liu [b,*], Jiancheng Lv [b,*]

[a] School of Computer Science and Artificial Intelligence, Wenzhou University, Wenzhou, PR China
[b] College of Computer Science, Sichuan University, Chengdu, PR China

ARTICLE INFO

Keywords:
Neural architecture search
Neural predictor
Active learning

ABSTRACT

Neural Predictors (NPs) are pivotal for efficient Neural Architecture Search (NAS) but suffer from limited accuracy due to the high cost of acquiring labeled training data. While active learning (AL) can mitigate this by prioritizing informative samples, existing methods suffer from selection bias under imbalanced data distributions, often sacrificing representativeness for diversity. To address this, we redefine the sample selection mechanism in AL and propose a Distribution-aware Active Learning framework for Neural Predictor (called ***DARE***). The goal is to select samples that not only ensure diversity but also exhibit a high degree of generalizability, making them more representative of the underlying data distribution. Our approach first extracts architecture representations via a graph-based encoder enhanced with a consistency-driven objective. Then, a two-stage selection strategy identifies both globally diverse and locally reliable samples through progressive representation learning and refinement. For non-uniform data distributions, we further introduce an adaptive mechanism that anchors sampling to key regions with high similarity density, avoiding performance degradation caused by outliers. Extensive experiments on NAS-Bench-101, NAS-Bench-201, NAS-Bench-301, and DARTS demonstrate that ***DARE*** achieves state-of-the-art prediction performance with extremely scarce labeled data. For instance, on NAS-Bench-101, it outperforms the second-best baseline by 4.29% in Kendall's Tau with only 424 samples. Furthermore, ***DARE*** efficiently identifies optimal architectures using as few as 100 queries on DARTS.

1. Introduction

In recent years, neural architecture search (NAS) (Elsken et al., 2019; Wu et al., 2025; Yu et al., 2025) has gained increasing attention as a powerful technique for automatically discovering optimal neural architectures. NAS has shown great promise in various domains, including but not limited to computer vision (CV) (Poyser & Breckon, 2024) and natural language processing (NLP) (Chen et al., 2024). However, a decent search capacity of traditional NAS often comes with a high cost in terms of time or computational resources. Therefore, it is urgent to design effective and reasonable acceleration strategies. Low-fidelity training, a common acceleration strategy in NAS, reduces evaluation time by shortening training epochs, dataset size, *etc.*, but may lead to inaccurate performance predictions and the omission of superior architectures due to insufficient training (Zhou et al., 2020).

* Corresponding authors.
E-mail addresses: yucy324@gmail.com (C. Yu), wangyifan5217@stu.scu.edu.cn (Y. Wang), tangchenwei@scu.edu.cn (C. Tang), juwei@scu.edu.cn (W. Ju), liuxianggen@scu.edu.cn (X. Liu), lvjiancheng@scu.edu.cn (J. Lv).

https://doi.org/10.1016/j.ipm.2026.104892
Received 14 December 2025; Received in revised form 26 March 2026; Accepted 5 May 2026
Available online 13 May 2026

An alternative to accelerate the process of NAS is leveraging a neural predictor (NP) to estimate the performance of neural architectures, obviating the high cost of model training in evaluation. Due to the superior characteristics of the NP, it soon won the attention of researchers. However, achieving precise evaluations from the NP requires substantial training samples, each involving training and testing over several hours or days (Deng et al., 2017). Therefore, considering the limited feasibility of acquiring a large number of labeled samples, there is an urgent need to extract the most informative samples from the existing data under resource constraints, thereby enhancing model performance. This raises the first major challenge: **(I)** ***How to effectively select highly informative architectures for training the predictor under a severely limited labeling budget?***

Given the pivotal role of training samples, Active Learning (AL) (He & Wei, 2026) techniques provide a new paradigm to address this challenge as an efficient optimization strategy. AL intelligently selects the most informative samples, achieving the greatest performance improvement with minimal labeling cost. However, current AL methods (Li et al., 2024; Lin et al., 2024) primarily focus on enhancing sample diversity, often overlooking the impact of data distribution characteristics on sample selection. As a result, the selected samples lack representativeness and fail to accurately reflect the distribution of the dataset, particularly evident in neural architecture datasets. This leads to the second key challenge: **(II)** ***How to balance diversity and distributional representativeness in the sampling process to ensure more effective predictor training?***

To solve both challenges, we redefine the sample selection mechanism in AL and propose a Distribution-aware Active Learning framework for Neural Predictor (called ***DARE***). The goal is to efficiently sample instances that exhibit diversity and a high degree of generalization while accounting for dataset-specific distributions. Specifically, in each active learning iteration, we first extract architecture embeddings using a graph-based encoder, trained with a consistency-preserving objective to improve representation quality. Based on these embeddings, we perform a two-stage sample selection process. The first stage identifies globally diverse candidates by computing pairwise distances between labeled and unlabeled architectures. The second stage enhances local reliability by leveraging proximity topology, such as clustering and Delaunay Triangulation calculation, to refine the neighborhood around labeled samples and assign pseudo-labels for retraining. Moreover, to address the common issue of non-uniform sample distributions in architecture spaces, we introduce an adaptive sampling mechanism that identifies anchor samples with strong coverage of the unlabeled pool and restricts sampling to informative intermediate-density regions. This ensures that the selected samples not only broaden the search space exploration but also align closely with the true data distribution, enabling the predictor to generalize effectively.

We evaluate our proposed ***DARE*** on three widely used NAS search spaces, *i.e.*, NAS-Bench-101 (Ying et al., 2019), NAS-Bench-201 (Dong & Yang, 2020), NAS-Bench-301 (Zela et al., 2022), and DARTS (Liu et al., 2018). The experimental results show that the NP achieves state-of-the-art prediction performance after training on samples selected by ***DARE***. Furthermore, the ***DARE*** also significantly improves the NAS performance in the search for the optimal neural architecture. Additionally, we validate the effectiveness of ***DARE*** on the TransNAS-Bench-101 (Duan et al., 2021) across various tasks, where it also achieves impressive performance. Finally, we also perform an in-depth analysis to verify the superiority of the proposed strategy. In summary, our contributions are: ❶ ***Problem Connection.*** Paying attention to the significance of training samples, we establish a novel connection between neural predictors and sampling bias, emphasizing that the quality and representativeness of training data are critical for reliable architecture performance estimation. This is the first work that focuses on the training data of neural predictors. ❷ ***Novel Framework.*** We introduce ***DARE***, a two-stage sampling framework that first selects diverse candidates via a max–min strategy, then adaptively refines the sampling regions using a key-point guided mechanism. This framework ensures both diversity and representativeness by explicitly connecting sampling behavior with data distribution. ❸ ***Comprehensive Validation.*** The experimental results show that ***DARE*** achieves state-of-the-art prediction performances under limited training data. In addition, we validate our method across different tasks and consistently achieve excellent performance.

2. Related wrok

2.1. Neural architecture performance predictors

Research on neural architecture performance predictors (NPs) has gained increasing attention, with existing approaches broadly categorized into learning curve-based (Ding et al., 2025) and model-based methods (Zhao et al., 2025). The former extrapolates final performance from partial training curves but suffers from instability and sensitivity to hyperparameters, often requiring multi-fidelity techniques (Falkner et al., 2018). Model-based methods are more widely adopted, where a regression model is trained directly on architecture representations, including traditional machine learning methods (Sun et al., 2019) and graph-based predictors (Ning et al., 2023; Shi et al., 2020). Recently, researchers have focused on enhancing the perception of graph structures through attention-based architectures. NN-Former (Xu et al., 2025) proposes an adjacent-sibling multi-head attention mechanism and a bidirectional graph isomorphism feed-forward network, rethinking the graph structure representation in neural architectures. SSRNAS (Y. Liu, Zhu, & Liu, 2021) proposes a method based on recursive attention and cell tensor-flow diagram to reduce the search space by pruning underperforming architectures and alleviating weight-sharing competition. Furthermore, the high cost of labeled training data has motivated studies on improving sample efficiency through semi-supervised learning (Tang et al., 2020) and data augmentation (Y. Liu, Tang, & Sun, 2021). For instance, SemiGAN-NAD (Liu et al., 2024) proposes a semi-supervised generative adversarial network method that captures the bidirectional joint distribution between architecture and performance through a Bi-Arch2Perf module, leveraging a large number of unlabeled architectures to infer high-performance architecture features with only a few labeled samples. Considering that these methods rely on the quality of raw labeled samples, and low-quality initial samples can significantly compromise their effectiveness, ***DARE*** addresses the problem from a data-centric perspective by leveraging active learning strategies to construct high-quality training samples. Compared to methods that focus on improving encoder architectures, ***DARE*** provides a general data selection strategy that complements existing predictors. Compared to semi-supervised methods, ***DARE***'s data selection strategy can serve as a preceding step to provide higher-quality initial distributions.

2.2. Active learning

Active learning (AL) (Chen et al., 2026; Kranjc et al., 2015) aims to select the most informative data for labeling to optimize model performance with minimal labeled data. Its strategies are mainly divided into uncertainty-based methods and diversity-based methods. The former typically selects samples with the lowest prediction confidence for labeling (Fuchsgruber et al., 2024), while the latter selects unlabeled samples that are least similar to the labeled ones, often combined with clustering algorithms (J. Li et al., 2024) to achieve similarity comparisons between samples (Tan et al., 2024). In recent years, AL has begun to be combined with NAS to address the adaptation problem between model capacity and data scale. Geifman and El-Yaniv (2019) proposed Active-iNAS, which employs incremental NAS within the active learning loop, dynamically increasing network depth to adapt to the growing training data, thereby maintaining the model's generalization ability. To address the issue that Active-iNAS requires training multiple candidate models, leading to excessively high computational costs, Active-SNDS (Zhang et al., 2023) proposes a structured neural depth search strategy that utilizes gradient descent based on structured variational inference to efficiently search for the optimal network depth during the active learning process. However, although these methods have made progress in the dynamic adjustment of model structure, at the sample selection level, existing diversity-based sampling strategies overlook the bias introduced by data distribution. Our study proposes an adaptive sampling strategy that ensures diversity while providing high-quality training data for NP.

3. Approach

3.1. Problem setting

NAS is to search from the space of neural architectures and identify the optimal one. Due to the high cost of the performance evaluation process in NAS, it is inevitable to design an efficient architecture evaluation method. To this end, we focus on accelerating this process with neural predictors.

Let the search space of the neural architectures be $\mathcal{X}$. Due to the limited budget, only parts of $\mathcal{X}$ are evaluated and obtain the ground truth $\mathcal{Y}$ (*i.e.*, the performance). Considering that there are tens of thousands of neural architectures in $\mathcal{X}$, it is not realistic to assign a true performance label to each network architecture, so the amount of data in $\mathcal{Y}$ is far less than $\mathcal{X}$, that is, $\mathcal{D} = \{\mathcal{X}; \mathcal{Y}\} = \{x_1, x_2, \dots, x_p; y_1, y_2, \dots, y_k\}$ $(k \ll p)$. Then, according to the performance in $\mathcal{Y}$, the neural architectures with the labels are obtained from $\mathcal{X}$, forming the training data $\mathcal{D}_{train} = \{\mathcal{X}^l; \mathcal{Y}\} = \{x_1^l, x_2^l, \dots, x_k^l; y_1, y_2, \dots, y_k\}$. The performance predictor P (a regression model) is trained with input $\mathcal{X}^l$, and the resulting output is compared with $\mathcal{Y}$. The objective function $J(\cdot)$ of this process can be formulated as:

$$J(\mathcal{W}, \mathscr{D}) = \frac{1}{|\mathscr{D}|} \sum_{i=1}^{|\mathscr{D}|} \mathcal{L}(P(\mathcal{W}, x_i), y_i), \tag{1}$$

where $\mathcal{W}$ is the training parameters of the regression model, $\mathscr{D}$ denotes the data involved in training, in this case $\mathcal{D}_{train}$, and $\mathcal{L}(\cdot)$ denotes the loss function of P. To get a well-performing predictor, it is necessary to put forward a high requirement for $\mathcal{X}^l$. Under the condition of keeping the number of samples unchanged, improving the quality of the samples is an effective way.

3.2. Overview

In this paper, we consider that the budget for annotating the samples is limited. We aim to effectively annotate the samples and thus obtain an accurate predictor for the final NAS. Formally speaking, let the budget of the annotation number be K. The problem of this paper is how to select these K samples from the search space, such that the performance predictor can be well-trained. To address the challenge, we propose a distribution-aware active learning framework to select the most informative K samples such that we can train an accurate NP based on them. Specifically, in each iteration of AL, we propose a two-stage max–min sampling strategy to ensure sample diversity (as illustrated in Fig. 1). In the first stage, the predictor extracts embeddings and computes distances between labeled and unlabeled samples, selecting those with the largest and smallest distances. In the second stage, spatial partitioning techniques such as clustering and Delaunay triangulation are employed to re-evaluate these distances and improve the reliability of the selected samples. To further handle the challenges introduced by non-uniform sample distributions, we incorporate a key-point guided adaptive sampling strategy that dynamically adjusts the sampling region, refining the candidate pool for more effective selection. The framework ensures that the queried samples are both diverse and well-aligned with the underlying distribution.

3.3. Two-stage max-min sampling

Calculating the distance between samples to enhance diversity is a commonly effective method. However, when an accurate representation of the samples cannot be ensured, distance calculations may not capture sample diversity. Therefore, in this section, a two-stage distance calculation is proposed to ensure diversity. In the first stage, we refined the model training process to extract more accurate sample representations, which were then used for distance calculation. In the second stage, a semi-supervised mechanism was employed to further train the model and recompute the distances, enhancing the reliability of the results.

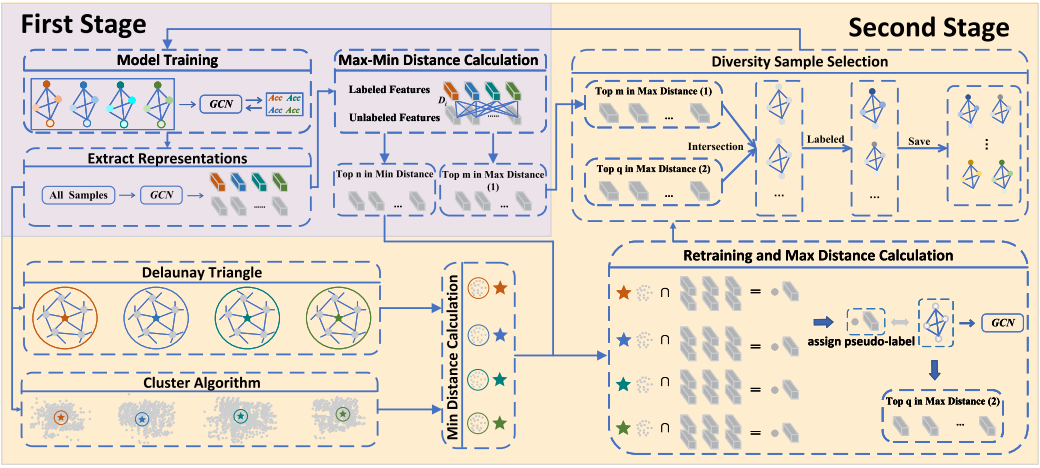

Fig. 1. The flowchart of the Two-stage Max-Min Sampling. First stage (purple background): (1) Model Training. (2) Max-Min Distance Calculation. The distance between labeled and unlabeled samples is calculated using the cosine distance. Second stage (yellow background): (1) Min Distance Calculation. The unlabeled samples with the minimum distance from the labeled samples are obtained by Delaunay triangulation or a clustering algorithm. (2) Retraining and Max Distance Calculation. The minimum distance between unlabeled samples obtained from the two acquisitions is intersected and assigned pseudo-labels provided to the model for retraining. (3) Diversity Sample Selection. Notably, the sample counts shown in this figure (e.g., "1") are for illustrative purposes only to simplify the workflow visualization. The actual number of selected samples in each step follows the parameters specified in Section 5.1 (Implementation Details).

First Stage: Model Training. The neural architecture is a model with extremely tight internal connections, and both the order of the structures and the way of connection are key factors affecting the performance, so representing it as a graph structure can highlight this characteristic. Consequently, a model capturing such a graph structure is essential for processing graph-structured data. In this context, employing graph convolutional networks (GCNs) (He & Wei, 2026; Li et al., 2025) enhances the extraction of meaningful representations $\mathbb{R}$ from neural architecture graphs. Specifically, we adopted the GCN to deal with the bi-directional information flow of the neural architecture (Wen et al., 2020). The layers (or blocks) of the neural architecture are represented by nodes I and transformed into a one-hot vector to form the operation matrix $\boldsymbol{O}$. In addition, the adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{I \times I}$ denotes the relationship among nodes. Therefore, the representations $\boldsymbol{V}$ of the sample at ith layer can be expressed as follows:

$$\begin{cases} \boldsymbol{V}_2 = \frac{1}{2}\,\mathrm{ReLU}\left(\boldsymbol{A}\boldsymbol{O}\boldsymbol{W}_1^{+}\right) + \frac{1}{2}\,\mathrm{ReLU}\left(\boldsymbol{A}^T\boldsymbol{O}\boldsymbol{W}_1^{-}\right) \\ \boldsymbol{V}_i = \frac{1}{2}\,\mathrm{ReLU}\left(\boldsymbol{A}\boldsymbol{V}_{i-1}\boldsymbol{W}_{i-1}^{+}\right) + \frac{1}{2}\,\mathrm{ReLU}\left(\boldsymbol{A}^T\boldsymbol{V}_{i-1}\boldsymbol{W}_{i-1}^{-}\right), \end{cases} \tag{2}$$

where $\boldsymbol{W}$ denotes the training parameters and RELU$(\cdot)$ denotes the activation function.

However, it is tough for the GCN to train with a small amount of data and obtain an accurate representation of the neural architecture, so we designed a new loss function to assist in training. Data augmentation (He & Wei, 2026; Y. Liu, Tang, & Sun, 2021) is an effective technique to increase the amount of data, especially in image processing. In this study, we perform data augmentation for each neural architecture and compare the similarity between the augmented and original architectures. As a result, we convert the loss function to:

$$\min_{\mathcal{W}} \frac{1}{k} \sum_{i=1}^{k} ((1-\lambda) \times \mathcal{L}(P(\mathcal{W}, x_i^l), y_i) + \lambda \times \mathcal{S}(x_{i,ori}^l, x_{i,aug}^l)), \tag{3}$$

where $x_{i,ori}^l$ and $x_{i,aug}^l$ denote the representation of the ith original and augmented sample, λ is the equilibrium coefficient, and k is the number of samples. In addition, $\mathcal{S}$ is the cosine distance.

First Stage: Max-Min Distance Calculation. Leveraging the trained model, we extracted the embeddings of all samples, including labeled and unlabeled ones, and computed the cosine distance D_i (i indicates the ith labeled data) between each labeled sample and all unlabeled samples. Then, $\mathcal{X}_i^{max} \in \mathcal{X}^u$ $(|\mathcal{X}_i^{max}| = m)$ and $\mathcal{X}_i^{min} \in \mathcal{X}^u$ $(|\mathcal{X}_i^{min}| = n)$ are extracted via D_i, denoting the m unlabeled samples furthest from the ith labeled sample and the n unlabeled samples nearest to ith the labeled sample, respectively. Crucially, the current computation method is better suited for uniformly distributed datasets. However, when dealing with non-uniformly distributed datasets (as illustrated in Fig. 2), the current maximum distance computation is not applicable (*c.f.*, Section 3.4).

Second Stage: Min Distance Calculation. For the initially obtained sets $\mathcal{X}_i^{max}$ and $\mathcal{X}_i^{min}$, their reliability and representativeness cannot be fully guaranteed due to potential noise and embedding inaccuracy. Therefore, a second refinement step is required to enhance selection precision, starting with a recalculation of the minimum-distance samples.

To improve the accuracy of nearest-neighbor selection, we further incorporate spatial partitioning methods to redefine sample neighborhoods from complementary perspectives: **(I) Delaunay Triangulation**. We perform Delaunay triangulation of the region

near the labeled samples, and an unlabeled sample node is considered neighbors (nearest samples) if it lies on the same side of at least one triangle as the nodes of the labeled samples. **(II) Cluster Algorithm**. The clustering algorithm is used to divide the unlabeled samples into L classes (L is the number of labeled samples) and make the labeled samples the centre of the clusters to find neighboring samples.

Based on the either method, $\mathcal{X}^{min*}{}_i \in \mathcal{X}^u(|\mathcal{X}^{min*}{}_i| = n^*)$ is obtained to denote the n^* nearest unlabeled samples to the ith labeled sample. Notably, the two methods are not used simultaneously. We prioritize Delaunay triangulation, as it captures the geometric structure of the sample space and yields more spatially coherent neighbors, making it well-suited for reliable distance refinement. However, due to its limited neighbor count, we also employ clustering to expand the neighborhood set and ensure sufficient overlap for subsequent intersection operations.

Second Stage: Retraining and Max Distance Calculation. Based on the nearest unlabeled samples obtained from the different rules above, we will first perform an intersection operation. Combining $\mathcal{X}_i^{min}$ and $\mathcal{X}^{min*}{}_i$, the samples coexisting in the two sets are acquired, *i.e.*, $\mathcal{X}_{i,comb}^{min} = \mathcal{X}_i^{min} \cap \mathcal{X}^{min*}{}_i$. Next, we assign pseudo-labels $\widetilde{\mathcal{Y}}$ to the $\mathcal{X}_{i,comb}^{min}$ and get the training data $D_{train}^* = \left\{\mathcal{X}_{1,comb}^{min}, \ldots, \mathcal{X}_{k,comb}^{min}; \widetilde{\mathcal{Y}}\right\}$ (k is the number of labeled samples in the labeled pool available). Finally, we use D_{train}^* to re-train the NP.

After re-training, the NP will extract the representations of all samples once again. Based on these new representations, we perform the maximum distance calculation and obtain $\mathcal{X}^{max*}{}_i \in \mathcal{X}^u \quad (|\mathcal{X}^{max*}{}_i| = m^*)$, *i.e.*, the furthest m^* unlabeled samples from the ith labeled sample (note that the difference of $\mathcal{X}_i^{max}$ and $\mathcal{X}^{max*}{}_i$ is that the unlabeled sample representations are different when participating in the distance calculation).

Second Stage: Diverse Sample Selection. Diversity selection aims to identify the samples with the lowest similarity between the obtained unlabeled samples and the existing labeled samples. So, we perform the following intersection of unlabeled samples:

$$\begin{cases} \mathcal{X}_{i,comb}^{max} = \left\{\mathcal{X}_i^{max} \cap \mathcal{X}^{max*}{}_i; i = 1, \ldots, K\right\} \\ \mathcal{X}_{final} = Top(\mathcal{X}_{i,comb}^{max}, n_s), \end{cases} \tag{4}$$

where the resulting $\mathcal{X}_{final}$ is the final selected unlabeled samples, which are also the least similar to the labeled sample. In addition, n_s denotes the number of samples selected in each iteration, and $Top(\cdot)$ denotes the n_s samples selected from $\mathcal{X}_{i,comb}^{max}$ with the greatest distance. Note that m^* is less than m to reduce the impact of pseudo-labeled samples.

3.4. Key-point guided adaptive sampling

The above two-stage max–min sampling strategy is designed for scenarios where samples are uniformly distributed in the embedding space. However, in practical neural architecture datasets, sample distributions often exhibit non-uniform characteristics (as illustrated in Fig. 2), where a large number of samples are concentrated in specific regions (green circles) while samples in peripheral areas are sparse and scattered. In such cases, directly applying the above strategy would cause the algorithm to overly focus on outlier samples that are far from the main distribution, thereby degrading performance.

To address this issue, we propose a key-point guided adaptive sampling strategy that replaces the maximum distance computation step used in uniform distribution scenarios, ensuring that the sampling process adapts to the true distribution characteristics of the data. The details are shown in the equations below:

$$\begin{cases} x_{key} = \text{argmin}_{x_i^l \in \mathcal{X}^l} \left(\frac{1}{T} \sum_{t=1}^{T} U(x_i^l)\right) \\ U(x_i^l) = \sum_{x_j^u \in \mathcal{X}^u} S\left(x_i^l, x_j^u\right), j = 1 \ldots z, \end{cases} \tag{5}$$

where x_{key} (*i.e.*, key point) is the labeled sample with the minimum average distance from z randomly selected unlabeled samples, T denotes the number of repetitions, $S(\cdot)$ represents the cosine distance, and $\mathcal{X}^u$ denotes the unlabeled data set. We assume that the resulting x_{key} has the highest proportion of similarity to unlabeled samples, which can cover more information about unlabeled samples. Then, we will select samples with the maximum distance between intervals $[|D_{x_{key}}| \times \alpha, |D_{x_{key}}| \times \beta]$ (the interval corresponds to the blue region in Fig. 2), where $D_{x_{\text{key}}}$ is the distance from x_{key} to all unlabeled samples (sorted from smallest to largest), α and β are coefficients and $\alpha < \beta$. Note that in a uniformly distributed scenario, the maximum distance calculation is performed for each labeled sample, whereas in a non-uniformly distributed scenario, the maximum distance is only calculated for key samples.

3.5. Complexity analysis

Time Complexity Analysis. To comprehensively evaluate the efficiency and scalability of the ***DARE*** framework, we provide a detailed analysis of the time complexity of its core components. Let N denote the size of the unlabeled sample pool, K denote the number of labeled samples (satisfying $K \ll N$), and d denote the dimension of architecture embeddings. First, in the architecture representation extraction stage, the time complexity of the GCN-based encoder for processing each architecture depends on the number of nodes and edges in the graph (denoted as a constant c) and the embedding dimension d, with single-sample inference complexity of $O(c \cdot d^2)$. Therefore, the feature extraction complexity for the entire search space is $O(N \cdot c \cdot d^2)$, which is linear with respect to the search space size. Second, in the two-stage max–min sampling process, computing pairwise cosine distances between K labeled samples and N unlabeled samples requires $O(K \cdot N \cdot d)$, and selecting the top m candidates involves sorting operations with complexity $O(K \cdot N \cdot \log m)$. The second-stage local refinement process, Delaunay triangulation and clustering algorithms are

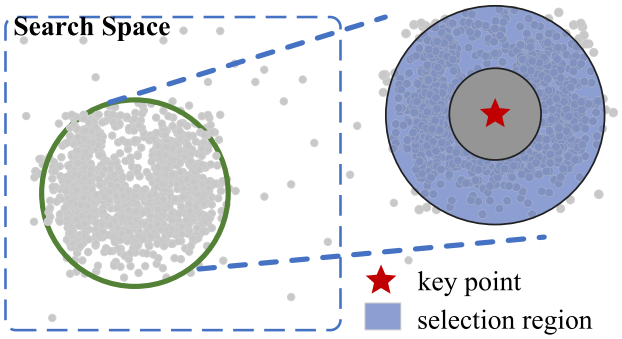

Fig. 2. The calculation of the maximum distance between unlabeled and labeled samples is performed in scenarios with a non-uniform distribution of similarity.

performed only on a very small candidate subset. Specifically, let n denote the size of the nearest-neighbor candidate set selected in the first stage (e.g., $n = 1000$, which is much smaller than the search space N). For Delaunay triangulation, we first reduce the high-dimensional embeddings to a low-dimensional space via PCA, and then perform triangulation with time complexity $O(n \log n)$. For clustering, the complexity is $O(n \cdot K \cdot I \cdot d)$, where K is the number of cluster centers (equal to the number of labeled samples) and I is the number of iterations. Since the candidate subset size $n \ll N$, and K, I, d are all small constants, the actual overhead of these two operations is negligible compared to the global distance computation ($O(K \cdot N \cdot d)$). Finally, the key-point guided adaptive sampling mechanism involves computing distances from the key point to all unlabeled samples ($O(N \cdot d)$) and sorting the distances to determine the sampling interval ($O(N \log N)$). In summary, the total time complexity per iteration of ***DARE*** is approximately $O(K \cdot N \cdot d + N \log N)$. Since K is much smaller than N and d is a constant, the overall complexity of the algorithm is dominated by the linear term in N, *i.e.*, $O(N)$. This linear scalability demonstrates that ***DARE*** can efficiently handle large-scale search spaces containing millions of architectures, making it highly suitable for practical neural architecture search scenarios.

Space Complexity Analysis. The memory consumption of ***DARE*** is primarily dominated by the storage of architecture embeddings and intermediate distance matrices, while the specific sub-modules (*e.g.*, Delaunay triangulation and clustering) are optimized to ensure low memory overhead. Storing embeddings for the entire search space requires $O(N \cdot d)$ space, and since the architecture encoder has a fixed parameter size independent of N, its storage overhead is negligible. Naively storing the pairwise distance matrix between K labeled and N unlabeled samples would require $O(K \cdot N)$ space; however, in our implementation, we utilize batch-wise computation, reducing the peak memory requirement to $O(B \cdot N)$ (where B is the batch size), ensuring scalability on standard GPUs. For Delaunay triangulation, the standard algorithm has a high space complexity of $O(n^{\lceil d/2 \rceil})$, which is prohibitive in high dimensions; however, in ***DARE***, we apply two crucial constraints: (1) it is performed only on a small subset of candidate samples n ($n \ll N$, e.g., $n = 1000$), and (2) we project these samples into a 2D subspace via PCA before triangulation, and consequently, the space complexity is strictly bounded to $O(n)$, making it extremely memory-efficient. For clustering (e.g., K-means), the operation is similarly restricted to the local candidate subset, and the space complexity for storing cluster centers and assignments is $O(n \cdot d + K \cdot d)$. Since n and K are small constants relative to N, this component introduces negligible memory overhead. In summary, the total space complexity of ***DARE*** is linear with respect to the search space size, *i.e.*, $O(N \cdot d)$, ensuring that ***DARE*** is scalable to large-scale benchmarks without memory bottlenecks.

4. Theoretical analysis

To achieve superior performance of the NP, we hope that the selected samples remain as diverse as possible while selecting as few samples as possible. On the other hand, we have found experimentally that the NP performs better when the distribution of the selected sample approximates the total sample. Proceeding from this, we provide an auxiliary theoretical intuition on why diverse and distributionally representative samples can improve sample efficiency in NP training. The details are shown in the Appendix.

Let $\mathcal{D}$ and $\mathcal{D}_{train}$ be the dataset containing all samples and the dataset of the selected samples (used as training data). $\cos(\cdot)$ denotes the cosine similarity calculation between samples. The following proposition gives the minimum value of n (*i.e.*, the number of selected samples) based on the cosine similarity between the samples.

Proposition 4.1. *Let* $\mathcal{D}_{train} = \{X_1, X_2, \ldots, X_n\}$, *and* $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$. *With probability of* $1 - \delta$, *we have:*

$$n \geq \sqrt{\frac{\sum_{i=1}^{n}(b_i - a_i)^2}{2t^2} \ln \frac{2}{\delta}}, \quad t = \frac{1}{n(n-1)} \sum_{i=0}^{n} \sum_{j \neq i}^{n} \left(1 - \cos\left(X_i, X_j\right)\right) + \epsilon, \tag{6}$$

where ϵ *is a very small positive number to avoid the situation that the denominator is 0 in Eq. (6).*

The proposition gives the bounded value of n, which is subject to the variable t. This means that the smaller the cosine similarity among the samples, the smaller the value of n, *i.e.* the higher the sample diversity and the smaller the number of samples to be selected. Notably, it does not directly derive the specific two-stage refinement or key-point interval selection used in DARE, but instead provides intuition for the importance of diversity and representativeness.

5. Experiments

5.1. Experimental setup

Datasets. This study mainly carries out experiments on search spaces, *i.e.*, NAS-Bench-101 (NB101) (Ying et al., 2019), NAS-Bench-201 (NB201) (Dong & Yang, 2020), NAS-Bench-301 (Zela et al., 2022), DARTS (Liu et al., 2018), and TransNAS-Bench-101 (TransBench-101) (Ying et al., 2019). The NAS-Bench-101 dataset contains 423K neural network architectures, each of which is trained, tested, and validated on CIFAR10 (Krizhevsky et al., 2009). The number of neural architectures in the NAS-Bench-201 is 15K, and each network architecture is trained on three image datasets, namely, CIFAR10, CIFAR100, and ImageNet16-120 (Russakovsky et al., 2015). NAS-Bench-301 is a surrogate search space comprising 10^{21} architectures. Each architecture consists of a normal cell and a reduction cell, each with seven nodes and eight edges. Following the settings in FlowerFormer (Hwang et al., 2024), we utilize a subset of 56,968 anchor architectures that are fully trained, employing their accuracy on the CIFAR-10 dataset for our experiments. DARTS is a method of differentiable architecture search that optimizes continuous parameters to find the best architecture. We tested our proposed method within the DARTS search space, which includes 7 nodes and 14 edges. In DARTS, each neural architecture is made up of two cells, a normal cell and a reduction cell. TransNAS-Bench-101 (Duan et al., 2021) is a multi-task neural architecture search (NAS) benchmark that provides performance data across seven tasks, including classification, regression, pixel-level prediction, and self-supervised learning. It features two types of search spaces: a macro-level space containing 3256 architectures, and a cell-level space with 4096 architectures. All architectures in both spaces have been trained, validated, and tested on the seven tasks.

Baselines. We compare ***DARE*** with multiple state-of-the-art methods. The competitors include: Peephole (Deng et al., 2017), E2Epp (Sun et al., 2019), SSANA (Tang et al., 2020), HAAP (Y. Liu, Tang, & Sun, 2021), RFGIAug (Xie et al., 2024), ReNAS (Xu et al., 2021), NPNAS (Wen et al., 2020), MLP (Wang et al., 2019), LSTM (Wang et al., 2019), BOGCN (Shi et al., 2020), HOP-2 (Z. Chen et al., 2021). In the architectural search for real scenarios, we also introduce additional comparison algorithms, including: GATES (Ning et al., 2023), NASBOT (White et al., 2020), ResNet (He et al., 2016), TNASP (Lu et al., 2021), PINAT (Lu et al., 2023), NAR-Former (Yi et al., 2023), BRP-NAS (Dudziak et al., 2020), MeCo (Jiang et al., 2024), SWAP (Peng et al., 2024), REA (Dong & Yang, 2020), RS (Dong & Yang, 2020), HNAS (Shu et al., 2022), and RoBoT (He et al., 2024).

Implementation Details and Evolution Metrics. All experiments are conducted on a server equipped with an NVIDIA GeForce RTX 3090 GPU (24 GB memory), an Intel Xeon Gold 5120 CPU @ 2.20 GHz (28 physical cores), and 754 GB of system memory. The software environment includes Python 3.11.7, PyTorch 2.1.0, and CUDA 11.8. We use a GCN model as the predictor, with the predictor's input the architecture's representation as multiple matrices, following the setting in Y. Liu, Tang, and Sun (2021). The selection in AL is divided into two parts, in which 5 samples are randomly selected for annotation at initialization, while 10 samples are subsequently selected at each iteration using the proposed sampling method. The predictor is trained using the Adam optimizer with an initial learning rate of 0.001 and a weight decay of 1e-5. The learning rate is scheduled using CosineAnnealingWarmRestarts with an initial restart period $T_0 = 20$ and a multiplier $T_{mult} = 2$, meaning the learning rate cosine-decays to zero within each cycle and then restarts, with cycle lengths of 20, 40, 80 epochs, and so on. No early stopping mechanism is employed, and all models are trained for a fixed number of 300 epochs. The batch size is the same as the number of samples selected at each iteration. The λ in Eq. (3) is 0.3. We evaluate the predictor using Kendall's Tau (KTau), Mean Squared Error (MSE), and Rank to assess ranking consistency and regression accuracy, while the top-1 architecture's performance is further reported using validation (Val) and test (Test) accuracy. In addition, for the ***DARE*** sampling strategy, the hyperparameters are adjusted according to the search space scale. For NAS-Bench-101, the size of the farthest candidate set m is set to 10,000, and the nearest candidate set n is 10,000 in the first stage. The refined farthest candidate size m^* is 10,000, while the key-point sampling parameters are set to $z = 1,000$ and $T = 10$. For NAS-Bench-201 and DARTS, these parameters are set to $m = 1,000$, $n = 1,000$, $m^* = 1,000$, $z = 500$, and $T = 10$.

5.2. Comparison results

We first compare the ***DARE*** with several SOTA methods on three datasets to verify its effectiveness. The comparison results are shown in Tables 1 and 2. ***DARE*** consistently outperforms all baseline methods across NB101, NB201, and NB301 datasets under various training sample sizes. On NB101, ***DARE*** achieves the highest ranking accuracy with a KTau improvement of up to 6.31% over the best baseline (HAAP) when using 1K training samples, while reducing the regression error (MSE) by 39.13%. Even with fewer samples (*e.g.*, $K = 424$), it still outperforms strong baselines such as NPNAS by 8.42% in KTau and 33.33% in MSE, demonstrating its effectiveness in low-resource settings. On NB201, ***DARE*** achieves a KTau of 0.7854 with $K = 150$, surpassing the second-best method (HAAP) by 6.49%, while maintaining the lowest MSE of 0.0005. When the sample size drops to 78, it still maintains a 1.05% lead in KTau over the next best performer, indicating that ***DARE*** selects more informative and generalizable samples under strict budget constraints. Similarly, on the NB301 dataset, ***DARE*** consistently outperforms advanced baselines such as FLOWERFORMER. Specifically, using only 1% of training samples, ***DARE*** achieves a KTau of 0.651, surpassing the runner-up by 0.9 points. This advantage becomes even more pronounced at the 5% sampling ratio, where ***DARE*** attains a KTau of 0.745, significantly leading the best baseline (FLOWERFORMER) by a margin of 2.3 points. This sustained superiority confirms that ***DARE*** can effectively identify high-quality architectures within vast search spaces, even under limited supervision.

Table 1
Comparison results of ***DARE*** with the SOTA methods on two datasets. "–" indicates the indicator could not be reproduced or has no available public report. Bold indicates the best result.

Methods	NAS-Bench-101			Methods	NAS-Bench-201		
	K	KTau↑	MSE↓		*K*	KTau↑	MSE↓
Peephole	1K	0.4373 ± 0.0112	0.0071 ± 0.0005	Peephole	150	0.5112 ± 0.0311	0.0062 ± 0.0010
E2Epp	1K	0.5705 ± 0.0082	0.0042 ± 0.0003		78	0.4561 ± 0.0280	0.0077 ± 0.0009
SSANA	1K	0.6541 ± 0.0078	0.0031 ± 0.0003	E2Epp	150	0.6699 ± 0.0100	0.0013 ± 0.0008
HAAP	1K	0.7126 ± 0.0024	0.0023 ± 0.0003		78	0.5729 ± 0.0193	0.0019 ± 0.0006
HAAP	424	0.7010 ± 0.0022	0.0024 ± 0.0003	HAAP	150	0.7375 ± 0.0200	**0.0005 ± 0.0001**
RFGIAug	424	0.6513 ± 0.0026	0.0019 ± 0.0002		78	0.6619 ± 0.0219	0.0011 ± 0.0003
ReNAS	424	0.6619 ± 0.0033	0.0021 ± 0.0005	RFGIAug	150	0.7219 ± 0.0019	0.0009 ± 0.0002
NPNAS	424	0.6743 ± 0.0029	0.0027 ± 0.0003		78	0.6941 ± 0.0008	0.0010 ± 0.0002
MLP	381	0.5116 ± 0.0011	0.0058 ± 0.0001	ReNAS	150	0.6731 ± 0.0041	0.0008 ± 0.0011
LSTM	381	0.5874 ± 0.0017	0.0046 ± 0.0002		78	0.6210 ± 0.0190	0.0012 ± 0.0005
BOGCN	381	0.5790	–	NPNAS	150	0.7004 ± 0.0031	0.0009 ± 0.0002
HOP-2	190	0.6440	–		78	0.6635 ± 0.0195	0.0010 ± 0.0017
DARE	1K	**0.7576 ± 0.0030**	**0.0014 ± 0.0001**	***DARE***	150	**0.7854 ± 0.0050**	**0.0005 ± 0.0002**
	424	0.7311 ± 0.0031	0.0018 ± 0.0002				
	381	0.6814 ± 0.0030	0.0018 ± 0.0001		78	0.7014 ± 0.0300	0.0008 ± 0.0003
	190	0.6593 ± 0.0030	0.0020 ± 0.0003				

Table 2
Comparison of predictive performance on NAS-Bench-301 using Kendall's Tau and Precision@Top K(%). Following the protocol in Hwang et al. (2024), results are averaged over 9 independent trials, involving three random seeds on three distinct splits, reported as mean ± standard deviation.

Datasets	NAS-Bench-301 (KTau ×100)			NAS-Bench-301 (P@Top K(%))		
Training/K	1%	5%	10%	1	5	10
GatedGCN (Bresson & Laurent, 2017)	61.8 ± 2.4	70.0 ± 0.9	71.4 ± 1.0	19.1 ± 4.1	55.2 ± 4.5	71.8 ± 2.9
DAGNN (Thost & Chen, 2021)	61.5 ± 1.9	70.9 ± 0.5	73.4 ± 1.2	23.1 ± 2.1	58.3 ± 3.4	73.1 ± 1.5
GraphGPS (Rampášek et al., 2022)	59.7 ± 1.8	69.3 ± 0.9	70.7 ± 1.2	20.6 ± 2.1	57.2 ± 3.8	73.4 ± 2.5
DAGFormer (Luo et al., 2023)	61.3 ± 2.0	70.7 ± 0.8	72.1 ± 0.8	20.7 ± 3.4	57.6 ± 3.7	73.4 ± 2.5
TA-GATES (Ning et al., 2022)	61.3 ± 1.2	68.9 ± 1.6	71.8 ± 1.6	20.1 ± 5.0	56.2 ± 6.1	72.4 ± 3.6
FLOWERFORMER (Hwang et al., 2024)	64.2 ± 1.6	72.2 ± 1.0	73.6 ± 1.3	20.8 ± 3.7	58.5 ± 2.5	74.7 ± 2.4
DARE	**65.1 ± 1.3**	**74.5 ± 1.1**	**73.6 ± 1.0**	**25.9 ± 1.8**	**60.3 ± 3.1**	**77.5 ± 2.8**

Table 3
Search results of ***DARE*** with the SOTA methods on NAS-Bench-101 and DARTS.

NAS-Bench-101				DARTS			
Methods	*K*	Accuracy(%)↑	Rank↓	Methods	*K*	CIFAR10 Accuracy(%)↑	ImageNet Accuracy(%)↑
Peephole	1K	93.41 ± 0.34	1922	TNASP	1000	97.48	75.50
E2Epp	1K	93.77 ± 0.13	687	PINAT	1000	97.58	77.80
SSANA	1K	94.01 ± 0.12	59	NAR-For	100	97.52	–
ReNAS	1K	93.95 ± 0.11	148	BRP-NAS	60	97.52	–
HAAP	1K	94.09 ± 0.11	16	MeCo	–	97.36	–
BOGCN	381	–	1362	SWAP	–	97.61	76.00
GATES	381	–	22	***DARE***	100	**97.63**	**78.01**
DARE	381	**94.11 ± 0.11**	**6**	***DARE***	60	97.55	77.13

5.3. Results of NAS

To test the performance of the ***DARE*** further, we apply the proposed algorithm to a real scenario to perform a search for the optimal network architecture. The search results in Tables 2, 3, and 4, demonstrate the strong generalization performance of architectures discovered by ***DARE*** across multiple benchmark datasets. On NB101, ***DARE*** achieves an accuracy of 94.11% with only 381 training samples, surpassing all other baselines, including HAAP and ReNAS, and ranking 6-*th* among all 423k architectures, which represents a substantial improvement in sample efficiency. On NB201, ***DARE*** achieves optimal results on all three image datasets. Compared to the *optimal*, the neural architectures obtained by ***DARE*** are all close or even equal in terms of accuracy. For example, on the CIFAR10 and CIFAR100 validation sets, ***DARE*** can search for the network architecture with the best performance, while on the CIFAR10 test set, the accuracy of the architecture searched by ***DARE*** differs from that of *optimal* by only 0.1%. On NB301, ***DARE*** exhibits superior search capability. As detailed in the P@Top K comparison, ***DARE*** achieves a precision of 25.9% for the Top 1 architectures, significantly outperforming the leading baseline FLOWERFORMER (20.8%) by 5.1%. This advantage is consistent across broader criteria, where ***DARE*** attains 60.3% and 77.5% for P@Top 5 and P@Top 10 respectively, verifying its

Table 4
Search results of ***DARE*** with SOTA algorithms on NAS-Bench-201.

Methods	CIFAR10		CIFAR100		ImageNet16-120	
	validation(%)	test(%)	validation(%)	test(%)	validation(%)	test(%)
RSPS	84.16 ± 1.69	87.66 ± 1.69	59.00 ± 4.60	58.33 ± 4.34	31.56 ± 3.28	31.14 ± 3.88
DATRS-V1	39.77 ± 0.00	54.30 ± 0.00	15.03 ± 0.00	15.61 ± 0.00	16.43 ± 0.00	16.32 ± 0.00
DARTS-V2	39.77 ± 0.00	54.30 ± 0.00	15.03 ± 0.00	15.61 ± 0.00	16.43 ± 0.00	16.32 ± 0.00
GDAS	90.00 ± 0.21	93.51 ± 0.13	71.15 ± 0.27	70.61 ± 0.26	41.70 ± 1.26	41.84 ± 0.90
ENAS	39.77 ± 0.00	54.30 ± 0.00	15.03 ± 0.00	15.61 ± 0.00	16.43 ± 0.00	16.32 ± 0.00
NPENAS	91.08 ± 0.11	91.52 ± 0.16	–	–	–	–
REA	91.19 ± 0.31	93.92 ± 0.30	71.81 ± 1.12	71.84 ± 0.99	45.15 ± 0.89	45.54 ± 1.03
RS	90.03 ± 0.36	93.70 ± 0.36	70.93 ± 1.09	71.04 ± 1.07	44.45 ± 1.10	44.57 ± 1.25
REINFORCE	91.09 ± 0.37	93.85 ± 0.37	71.61 ± 1.12	71.71 ± 1.09	45.05 ± 1.02	45.24 ± 1.18
BOHB	90.82 ± 0.53	93.61 ± 0.52	70.74 ± 1.29	70.85 ± 1.28	44.26 ± 1.36	44.42 ± 1.49
NASBOT	–	93.64 ± 0.23	–	71.38 ± 0.82	–	45.88 ± 0.37
E2Epp	90.61 ± 0.89	93.39 ± 0.75	71.08 ± 2.00	71.11 ± 1.93	44.36 ± 1.85	44.87 ± 1.43
HAAP	91.18 ± 0.25	94.00 ± 0.25	71.24 ± 1.48	71.58 ± 1.56	45.31 ± 1.14	46.03 ± 0.90
ReNAS	90.90 ± 0.31	93.99 ± 0.25	71.96 ± 0.99	72.12 ± 0.79	45.85 ± 0.47	45.97 ± 0.49
DARE	**91.47 ± 0.14**	**94.06 ± 0.30**	**72.1 ± 1.39**	**72.53 ± 0.51**	**45.90 ± 0.65**	**46.47 ± 0.23**
ResNet	90.83	93.97	70.42	70.86	44.53	43.63
optimal	91.61	94.37	73.49	73.51	46.77	47.31

Table 5
Search results on TransNAS-Bench-101. Bold is the optimal value.

Space	Methods	Accuracy (%)			L2 Loss ($\times 10^{-2}$)	mIoU (%)	SSIM ($\times 10^{-2}$)	
		Scene	Object	Jigsaw	Layout	Segment.	Normal	Autoenco.
Micro	REA	54.63	44.92	94.81	**−62.06**	94.55	57.10	56.23
	RS	54.56	44.76	94.63	−62.22	94.53	56.83	55.55
	HNAS	54.29	44.08	94.56	−64.83	94.57	56.88	48.66
	RoBoT	54.87	45.59	94.76	−62.12	94.58	57.44	55.30
	DARE	**54.91**	**45.59**	**95.01**	-62.34	**94.61**	**58.03**	**56.33**
	Optimal	54.94	45.59	95.37	−60.10	94.61	58.73	57.72
Macro	REA	56.69	46.74	96.78	−59.99	94.80	60.81	71.38
	RS	56.53	46.68	96.74	−60.27	94.78	60.72	70.05
	HNAS	55.03	45.00	96.28	−61.40	94.79	59.27	57.59
	RoBoT	57.41	46.87	96.89	−58.44	94.83	61.66	73.51
	DARE	**57.41**	**47.03**	**97.02**	**−58.22**	**94.86**	**62.17**	**73.89**
	Optimal	57.41	47.42	97.02	−58.22	94.86	64.35	74.88

ability to accurately locate high-quality architectures in vast search spaces. On the DARTS search space, due to the vast search space and complex downstream tasks, we have carried out the following operations. First, we use ***DARE*** to select training samples and train the neural predictor on the DARTS search space, and search for the optimal cell structure based on the trained predictor. Then, we train the complete network from scratch on CIFAR-10 using the discovered cell structure and evaluate its performance. Finally, we directly transfer the searched architecture to ImageNet, train it from scratch on ImageNet, and evaluate its Top-1 accuracy. The result shows ***DARE*** also achieves state-of-the-art performance with only 100 queried samples, reaching 97.63% on CIFAR-10 and 78.01% on ImageNet, outperforming existing NAS approaches such as TNASP, PINAT, and SWAP. These results collectively verify the effectiveness of ***DARE*** in identifying high-performing architectures under limited supervision across diverse search spaces.

5.4. Application on various tasks

To validate the effectiveness of our sampling strategy, we further conduct experiments on various tasks from TransBench-101 (Micro). Table 5 presents the comparison on TransNAS-Bench-101 across both Micro and Macro search spaces. ***DARE*** consistently outperforms state-of-the-art methods like REA and RS on the majority of tasks. In the Micro space, our method matches the *Optimal* architecture in Object and Segmentation tasks. While slightly outperformed by REA on the Layout task in the Micro setting, ***DARE*** surpasses all competitors in the Macro space, remarkably achieving *Optimal* performance on four tasks (Scene, Jigsaw, Layout, and Segmentation). These results confirm the superior generalization capability of ***DARE*** across diverse non-classification tasks.

5.5. Ablation study

We conduct ablation studies on both NAS-Bench-101 and NAS-Bench-201 to comprehensively assess the contributions of each component in our framework. Specifically, ***DARE*** w/o FS or SS denotes the removal of the first-stage or second-stage max–min

Table 6
Ablation Study on NB101 and NB201.

Distribution	NAS-Bench-101		NAS-Bench-201	
	Uniform	Non-Uniform	Uniform	Non-Uniform
w/o FS	0.6628	0.6518	0.7172	0.6634
w/o SS	0.6564	0.6834	0.7059	0.6791
w/o TMMS	0.6127	0.6255	0.5743	0.6118
w/o KGA	0.6943	0.7122	0.7258	0.7029
DARE	**0.7019**	**0.7311**	**0.7714**	**0.7854**

Table 7
Comparison of active learning strategies on NB101.

Methods	KTau	MSE
N.S.	0.5817±0.0055	0.0024±0.0005
EOAL	0.6472±0.0035	0.0020±0.0003
BAL	0.6351±0.0072	**0.0014±0.0003**
DARE	**0.7311±0.0031**	0.0018±0.0002

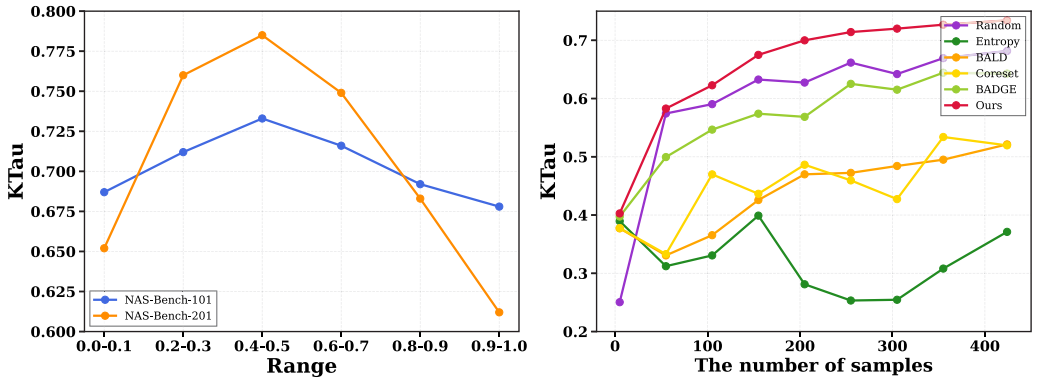

Fig. 3. Left: Sensitive analysis of α and β. **Right**: Performance comparison curves of different sampling strategies on NAS-Bench-101.

sampling strategy, respectively. ***DARE*** w/o TMMS represents the exclusion of the entire two-stage max–min sampling strategy, while ***DARE*** w/o KGA indicates removing the key-point guided adaptive sampling strategy. Note that KGA is dependent on the existence of TMMS.

As shown in Table 6, the results are consistent across both datasets. First, removing either the first stage (w/o FS) or the second stage (w/o SS) leads to a noticeable decline in model performance. Second, the removal of TMMS results in a sharp performance drop in all settings, highlighting its critical role in ensuring sample diversity and representativeness. Finally, the impact of removing KGA is consistently more pronounced under non-uniform distributions compared to uniform ones. This demonstrates that KGA effectively adapts the sampling region to account for data imbalances, thereby enhancing performance in complex data environments. Ultimately, the ***DARE*** framework, which integrates all the aforementioned components, consistently achieves the best performance across both datasets and distribution settings, validating the effectiveness and synergy of our proposed design.

5.6. Sensitive analysis

The sensitivity analysis illustrates the impact of the sampling range (*i.e.,* α and β) in Key-point Guided Adaptive Sampling. As shown in Fig. 3(Left), the choice of range has a significant impact on model performance. When the selected range is centered (*i.e.*, 0.4–0.5), the model achieves the highest KTau on both datasets, indicating that samples in this region are more representative. In contrast, when the range is too narrow and close to the key point (*i.e.*., 0.0–0.1), the selected samples tend to be redundant and offer limited additional information. On the other hand, selecting from a range too far from the key point (*i.e.*, 0.9–1.0) leads to performance degradation as these samples may lie in sparse or noisy regions of the space, reducing the reliability of the predictor. These findings highlight the importance of setting the sampling interval to balance informativeness and stability.

5.7. Expansion experiments on different active learning methods

To comprehensively evaluate the effectiveness of our active learning strategy, we conducted a fair comparison against several state-of-the-art methods, including NoiseStability (N.S.) (Li et al., 2024), EOAL (Safaei et al., 2024), and BAL (J. Li et al., 2024),

Table 8
Performance on different samples.

Predictors	selected	random
LR	**0.3969**	0.35042
RF	**0.5674**	0.51668
AdaBoost	**0.3917**	0.33342
Bagging	**0.4918**	0.46574
ExtraTree	**0.3541**	0.31348

Table 9
Performance on transformer-base structure.

Method	Small (Acc. (%)/Params)	Base (Acc. (%)/Params)
AutoFormer	81.7/22.9M	82.4/54.4M
TF-NAS	81.9/23.0M	82.2/56.5M
AZ-NAS	82.2/23.8M	82.3/54.1M
DARE	**83.1/22.7M**	**82.4/53.8M**

Table 10
Comparison of computational efficiency metrics on NAS-Bench-101. Parameters (K), FLOPs (M), and Inference Speed (ms) are reported for the top-1 architectures discovered by each method. Bold indicates the best result.

Metric	Parameters (M)	FLOPs (M)	Inference Speed (ms)
HAAP	0.10	0.67	0.93
NAR-Former	4.82	27.25	9.92
PINAT	0.55	3.77	5.30
DARE	**0.08**	**0.65**	**0.84**

on the NAS-Bench-101 dataset. In the experiment, neural predictors were trained using 424 samples, with KTau and MSE as the evaluation metrics. As shown in Table 7, the results strongly demonstrate the superiority and robustness of our method. It achieved a KTau of 0.7311, significantly surpassing all baselines and showcasing a superior ability to rank architectural performance. Meanwhile, although its MSE (0.0018) is slightly higher than that of BAL (0.0014), it remains highly competitive. Moreover, its substantial advantage in KTau confirms that its overall performance is optimal for tasks that rely on precise ranking, such as neural architecture search.

Analysis. Although N.S., EOAL, and BAL represent the latest advancements in the field of active learning, their performance in the NAS scenario still lags behind ***DARE***, which is intrinsically linked to the design mechanisms of each method. N.S. relies on small perturbations of model parameters to measure feature deviation for uncertainty assessment. However, in neural architecture performance prediction (a regression task), samples that are highly sensitive to parameter noise often correspond to architectures with unstable training or poor convergence. Consequently, N.S. tends to select these 'unstable' samples, inadvertently introducing low-quality noisy data rather than informative 'hard examples' for ranking. EOAL is designed specifically for open-set scenarios, utilizing dual entropy scores and virtual OOD prototypes to filter samples of unknown classes. Yet, NAS is a closed-set regression task where the concept of 'unknown classes' does not exist, rendering its core OOD detection mechanism ineffective in this context. BAL dynamically balances diversity and uncertainty via self-supervised features and the cluster distance difference metric. However, its adaptive sub-pool strategy assumes that dense regions in the feature space represent highly representative samples. This assumption fails under the long-tail distribution of NAS, where high-performance architectures are not located in feature-dense regions but are concentrated in specific performance intervals. In contrast, ***DARE*** constructs an embedding space via performance supervision, ensuring distance calculations directly reflect performance discrepancies among architectures. Meanwhile, the key-point guided mechanism explicitly identifies high-similarity density regions and samples around them, fundamentally adapting to the distributional characteristics of NAS datasets.

5.8. Expansion experiments on Transformer-based structure

To validate the effectiveness of our method in large-scale Transformer search spaces, we conducted experiments on the AutoFormer search space and compared it against baselines, including AutoFormer (M. Chen et al., 2021), TF-NAS (Zhou et al., 2022), and AZ-NAS (Lee & Ham, 2024) (Table 9). The results clearly demonstrate ***DARE***'s strong competitiveness. In the Small model comparison, ***DARE*** achieved the highest classification accuracy (83.1%) with the fewest parameters (22.7M). Similarly, in the larger Base model category, ***DARE*** achieved the highest accuracy on par with the baseline (82.4%) while maintaining the best parameter efficiency (53.8M). These results strongly prove that our method can be successfully extended to complex Transformer architecture search tasks, consistently discovering superior model architectures that balance both high accuracy and efficiency.

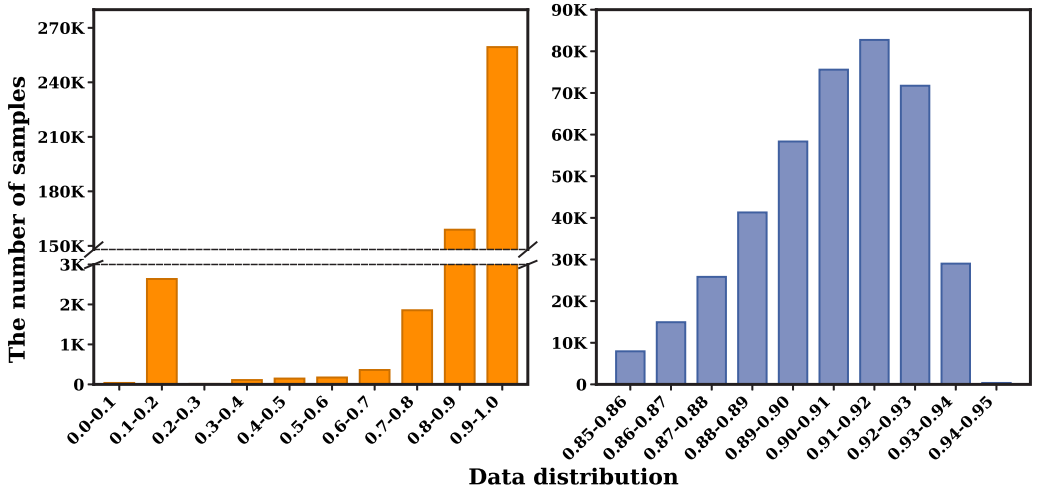

Fig. 4. Data distribution based on test accuracy in NAS-Bench-101. The x-axis coordinate indicates the accuracy range, *e.g.*, "0.0–0.1" means that the test accuracy of the sample (network architecture) is between [0.0, 0.1], and the y-axis coordinate indicates the number of samples in the corresponding accuracy range. The left shows the distribution of all samples in the NAS-Bench-101, and the right shows the distribution of samples with test accuracy between [0.85, 0.95].

5.9. Other experiments

Sampling Strategies. To verify the effectiveness of the proposed strategy, we compare it with other AL methods, including: Random, Entropy (Wang & Shang, 2014), BALD (Gal et al., 2017), CoreSet (Sener & Savarese, 2018), BADGE (Ash et al., 2020). These methods select 424 samples on NAS-Bench-101. We record the change in KTau during the acquisition process. As shown in Fig. 3(Right), the samples obtained by our strategy resulted in a leading performance of the predictor from the beginning to the end. In addition, our strategy improved by at least 0.05 in KTau compared to other strategies, while compared with Entropy, it has improved by more than 0.3. This demonstrates the effectiveness of our proposed strategy for performance predictors. However, there is an oddity in this experiment, *i.e.*, the Random method works better than other strategies. This is due to a data distribution problem with NAS-Bench-101, which we will detail in the **In-depth Analysis**.

Neural Predictors. To verify the generalizability of our proposed strategy, we have tested the selected samples using several predictors, including Linear Regression (LR), Random Forest (RF) (Breiman, 2001), AdaBoost (Freund & Schapire, 1997), Bagging (Breiman, 1996), and ExtraTree (Geurts et al., 2006). In addition, we use randomly selected samples as controls. As shown in Table 8, "selected sample" denotes the samples selected by our strategy, and "random samples" denotes the randomly selected samples. The sample sizes are both 424. In the table, all predictors perform better on samples obtained by our strategy than on samples obtained at random, which also proves the generalizability of our proposed algorithm.

Computational Efficiency. To further evaluate the practical applicability of the architectures discovered by ***DARE***, we compare the computational efficiency of the top-1 architectures found by different methods on NAS-Bench-101. As shown in Table 10, ***DARE*** discovers the most efficient architecture across all metrics. Specifically, the architecture identified by ***DARE*** contains only 0.08M parameters and requires 0.65M FLOPs, both of which are the lowest among all compared methods. In terms of inference speed, ***DARE*** achieves 0.84 ms per sample, outperforming HAAP (0.93 ms), NAR-Former (9.92 ms), and PINAT (5.30 ms) by a considerable margin. These results indicate that the distribution-aware sampling strategy of ***DARE*** not only enables the predictor to accurately identify high-performance architectures but also favors architectures with efficient designs. This further validates the practical value of ***DARE*** in real-world NAS scenarios where both accuracy and efficiency are essential.

5.10. In-depth analysis

In this section, we analyze the sample distribution in the NAS-Bench-101 and explain the question raised in **Sampling strategies** why Random would be more effective than other sampling strategies. Initially, we analyze the data distribution in the NAS-Bench-101. In Fig. 4, we plot the distribution of the data based on test accuracy, where the x and y axes indicate the accuracy range and the corresponding number of samples, respectively. As can be visualized in Fig. 4(left), although the range of accuracy is [0.0, 1.0], the number of samples with an accuracy greater than 0.8 accounts for nearly 99% of the total sample, which is the non-uniform distribution scenario. After further analysis, we find that the samples' accuracy is mainly concentrated between [0.85, 0.95], and therefore a more refined distribution is plotted in Fig. 4(right). Based on the above analysis, we can obtain that to train an NP with excellent performance, the training samples should be selected in an accuracy range of [0.85, 0.95]. Thus, as depicted in Fig. 5, the data distributions of the various sampling strategies are displayed. In Fig. 5, the histograms show the number of samples. The red line and the orange line are both proportional curves (see the legend in Fig. 5 for details).

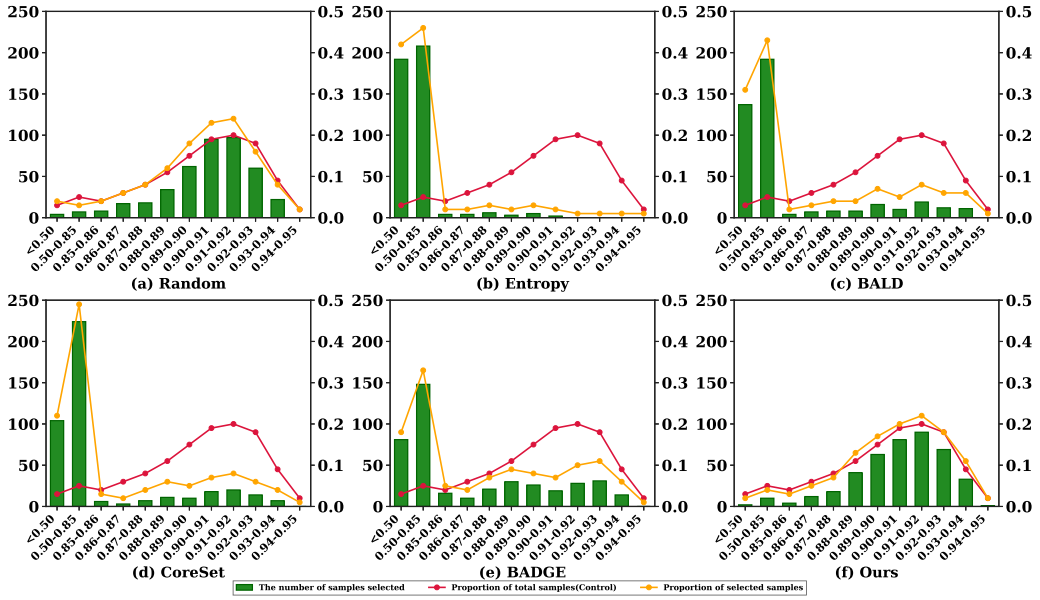

Fig. 5. Distribution of samples selected by different sampling strategies. The x-axis coordinate indicates the accuracy range, the y-axis coordinate (left) indicates the number of samples selected by the sampling strategy in different accuracy ranges, and the y-axis coordinate (right) indicates the proportion of samples to the total samples. Specifically, the histogram is the number of samples. The red line refers to the proportion of the number of samples in the corresponding accuracy range to the total number of samples in the NAS-Bench-101. The orange line refers to the proportion of the number of samples selected by the sampling strategy in the corresponding accuracy range to the total number of samples selected (the total number is 424).

Combining Fig. 5 with Fig. 3(Right), it is clear that the performance of the NP is related to two factors, the number of samples selected in the accuracy range of [0.85,0.95] and the proportion of samples selected, both of which our proposed algorithm is optimal in. On the other hand, the main reason for the poor results of Entropy, BALD, CoreSet, and BADGE is that too few samples are selected in the accuracy range [0.85, 0.95] and too many samples are selected outside the range, resulting in an imbalance in the training samples. In addition, as Random is free from human interference, the likelihood of selecting a sample is proportional to its frequency in the dataset. Thus, a large number of samples in the range [0.85, 0.95] increases the probability of selecting samples from this region, which coincidentally enhances the quality of the training samples and hence the predictor performance.

6. Conclusion and future work

This study highlights the critical role of training samples in the training process of neural predictors and proposes a novel and significant distribution-aware active learning method tailored for neural architecture datasets. The method incorporates a two-stage max–min selection strategy to ensure the diversity of selected samples and introduces a key-point-guided adaptive sampling strategy to enhance their representativeness, thereby comprehensively improving sample quality. Extensive experiments across various datasets validate the effectiveness and advantages of the proposed approach.

Currently, ***DARE*** adopts a static batch sampling mode, where sample selection is performed based on fixed labeled and unlabeled pools in each AL iteration. In the future, we plan to extend ***DARE*** to online learning and incremental learning scenarios by introducing a dynamic sampling mechanism, enabling the framework to adaptively adjust the sampling strategy based on real-time feedback from the predictor, achieving dynamic updating of training samples and continuous optimization of predictors to better adapt to the evolving demands and the continuous emergence of new architectures in practical neural architecture search applications.

CRediT authorship contribution statement

Caiyang Yu: Writing – original draft, Validation, Methodology, Conceptualization. **Yifan Wang:** Validation. **Chenwei Tang:** Writing – review & editing, Methodology. **Wei Ju:** Writing – review & editing. **Xianggen Liu:** Methodology. **Jiancheng Lv:** Validation.

Acknowledgment

This work is supported by the National Major Scientific Instruments and Equipments Development Project of National Natural Science Foundation of China under Grant 62427820.

Appendix. Theoretical analysis

To achieve superior performance of the NP, we hope that the selected samples remain as diverse as possible while selecting as few samples as possible. On the other hand, we have found experimentally (**In-depth Analysis**) that the NP performs better when the distribution of the selected sample approximates the total sample. Proceeding from this, we theoretically analyze the minimum number of selected samples when the distributions are similar.

We first introduce Lemma 1 (Hoeffding's inequality Hoeffding, 1994), which is a theorem in probability theory, and further, we deduce the minimum number of selected samples to be taken.

Lemma 1 (*Hoeffding, 1994*). *Let $X_1, X_2, \dots, X_n$ be a collection of n independent random variables, each with support in the intervals $[a_i, b_i]$, and let the expected value be $\mu = \frac{1}{n}\sum_{i=1}^{n} E[X_i]$. For any $t > 0$, it holds that:*

$$P\Big(\Big|\frac{1}{n}\sum_{i=1}^{n} X_i - \mu\Big| \geq t\Big) \leq 2\exp\Big(\frac{-2n^2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\Big). \tag{A.1}$$

Proposition 1. *Let $D_{train} = \{X_1, X_2, \dots, X_n\}$, and $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$. With probability of $1 - \delta$, we have:*

$$n \geq \sqrt{\frac{\sum_{i=1}^{n}(b_i - a_i)^2}{2t^2} \ln\frac{2}{\delta}}, \tag{A.2}$$

and

$$t = \frac{1}{n(n-1)}\sum_{i=0}^{n}\sum_{j\neq i}^{n}\big(1 - \cos\big(X_i, X_j\big)\big) + \epsilon. \tag{A.3}$$

Let $\mathcal{D}$ and $\mathcal{D}_{train}$ be the dataset containing all samples and the dataset of the selected samples (used as training data). We can translate the bias calculations for the variables in the Lemma into differences in sample distributions. For n samples $X_1, X_2, \dots, X_n$, the sample mean is $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$, and the sample variance is $S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2$. Then Hoeffding's inequality can be written in the following form:

$$P\Big(\big|\bar{X} - \mu\big| \geq t\Big) \leq 2\exp\Big(\frac{-2n^2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\Big), \tag{A.4}$$

where μ represents the excepted value in $\mathcal{D}$, and $\sum_{i=1}^{n}(b_i - a_i)^2$ is the upper and lower bound of all sample values.

We can solve the inequality about n by restricting the value of the upper bound on the probability of the right-hand side of the inequality. Specifically, assuming we want the upper bound of the probability to be less than probability δ, that is:

$$2\exp\Big(-\frac{2n^2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\Big) \leq \delta. \tag{A.5}$$

By transformation, we have:

$$n \geq \sqrt{\frac{\sum_{i=1}^{n}(b_i - a_i)^2}{2t^2} \ln\frac{2}{\delta}}. \tag{A.6}$$

Therefore, when we know the sample value and the probability upper bound δ we want, we can use the above formula to calculate the minimum sample size n that meets the requirement of the probability upper bound.

In addition, the prerequisite for achieving an approximation of the sample distribution is to ensure that the sample is diverse, so it is necessary to combine the difference value t with the sample diversity, which has:

$$t = \frac{1}{n(n-1)}\sum_{i=0}^{n}\sum_{j\neq i}^{n}\big(1 - \cos\big(X_i, X_j\big)\big) + \epsilon, \tag{A.7}$$

where $\cos(\cdot)$ is used to calculate the cosine similarity between samples and ϵ is a very small positive number to avoid the situation that the denominator is 0 in Eq. (A.6).

Data availability

No data was used for the research described in the article.

References

Ash, J. T., Zhang, C., Krishnamurthy, A., Langford, J., & Agarwal, A. (2020). Deep batch active learning by diverse, uncertain gradient lower bounds. In *International conference on learning representations*.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*, 123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5–32.

Bresson, X., & Laurent, T. (2017). Residual gated graph convnets. arXiv preprint arXiv:1711.07553.

Chen, A., Dohan, D., & So, D. (2024). EvoPrompting: language models for code-level neural architecture search. In *Advances in neural information processing systems*: *vol. 36*.

Chen, M., Peng, H., Fu, J., & Ling, H. (2021). Autoformer: Searching transformers for visual recognition. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 12270–12280).

Chen, Z., Zhan, Y., Yu, B., Gong, M., & Du, B. (2021). Not all operations contribute equally: Hierarchical operation-adaptive predictor for neural architecture search. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 10508–10517).

Chen, Y., Zhou, T., Zeng, D., Li, S., Liu, K., & Zhao, J. (2026). ASDE: Low-budget text classification via active semi-supervised learning with debiasing training mechanism. *Information Processing & Management*, *63*(2), Article 104390.

Deng, B., Yan, J., & Lin, D. (2017). Peephole: Predicting network performance before training. arXiv preprint arXiv:1712.03351.

Ding, Y., Huang, Z., Shou, X., Guo, Y., Sun, Y., & Gao, J. (2025). Architecture-aware learning curve extrapolation via graph ordinary differential equation. In *Proceedings of the AAAI conference on artificial intelligence*: *vol. 39*, (pp. 16289–16297).

Dong, X., & Yang, Y. (2020). NAS-bench-201: Extending the scope of reproducible neural architecture search. In *International conference on learning representations*.

Duan, Y., Chen, X., Xu, H., Chen, Z., Liang, X., Zhang, T., & Li, Z. (2021). Transnas-bench-101: Improving transferability and generalizability of cross-task neural architecture search. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 5251–5260).

Dudziak, L., Chau, T., Abdelfattah, M., Lee, R., Kim, H., & Lane, N. (2020). Brp-nas: Prediction-based nas using gcns. In *Advances in neural information processing systems*: *vol. 33*, (pp. 10480–10490).

Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural architecture search: A survey. *Journal of Machine Learning Research*, *20*(55), 1–21.

Falkner, S., Klein, A., & Hutter, F. (2018). BOHB: Robust and efficient hyperparameter optimization at scale. In *International conference on machine learning* (pp. 1437–1446). PMLR.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139.

Fuchsgruber, D., Wollschläger, T., Charpentier, B., Oroz, A., & Günnemann, S. (2024). Uncertainty for active learning on graphs. In *Proceedings of the 41st international conference on machine learning* (pp. 14275–14307).

Gal, Y., Islam, R., & Ghahramani, Z. (2017). Deep bayesian active learning with image data. In *International conference on machine learning* (pp. 1183–1192). PMLR.

Geifman, Y., & El-Yaniv, R. (2019). Deep active learning with a neural architecture search. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc.

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, *63*, 3–42.

He, Z., Shu, Y., Dai, Z., & Low, B. K. H. (2024). Robustifying and boosting training-free neural architecture search. In *The twelfth international conference on learning representations*.

He, L., & Wei, Z. (2026). Towards structure-aware data augmentation for high-degree graph neural networks. *Information Processing & Management*, *63*(1), Article 104343.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Hoeffding, W. (1994). Probability inequalities for sums of bounded random variables. *The Collected Works of Wassily Hoeffding*, 409–426.

Hwang, D., Kim, H., Kim, S., & Shin, K. (2024). Flowerformer: Empowering neural architecture encoding using a flow-aware graph transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 6128–6137).

Jiang, T., Wang, H., & Bie, R. (2024). MeCo: Zero-shot NAS with one data and single forward pass via minimum eigenvalue of correlation. In *Advances in neural information processing systems*: *vol. 36*.

Kranjc, J., Smailović, J., Podpečan, V., Grčar, M., Žnidaršič, M., & Lavrač, N. (2015). Active learning for sentiment analysis on data streams: Methodology and workflow implementation in the ClowdFlows platform. *Information Processing & Management*, *51*(2), 187–203.

Krizhevsky, A., Hinton, G., et al. (2009). *Learning multiple layers of features from tiny images*. Toronto, ON, Canada.

Lee, J., & Ham, B. (2024). Az-nas: Assembling zero-cost proxies for network architecture search. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 5893–5903).

Li, J., Chen, P., Yu, S., Liu, S., & Jia, J. (2024). Bal: Balancing diversity and novelty for active learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *46*(5), 3653–3664.

Li, H., Liang, H., Hu, Y., & Liu, X. (2025). Technology convergence prediction based on temporal heterogeneous graph neural networks. *Information Processing & Management*, *62*(3), Article 104034.

Li, X., Yang, P., Gu, Y., Zhan, X., Wang, T., Xu, M., & Xu, C. (2024). Deep active learning with noise stability. In *Proceedings of the AAAI conference on artificial intelligence*: *vol. 38*, (12), (pp. 13655–13663).

Lin, J., Liang, Z., Deng, S., Cai, L., Jiang, T., Li, T., Jia, K., & Xu, X. (2024). Exploring diversity-based active learning for 3d object detection in autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, *25*(11), 15454–15466.

Liu, H., Simonyan, K., & Yang, Y. (2018). DARTS: Differentiable architecture search. In *International conference on learning representations*.

Liu, Y., Tang, Y., & Sun, Y. (2021). Homogeneous architecture augmentation for neural predictor. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 12249–12258).

Liu, Y., Yu, Z., Liu, Z., & Tian, W. (2024). Efficient neural architecture design via capturing architecture-performance joint distribution. In *International conference on artificial intelligence and statistics* (pp. 1738–1746). PMLR.

Liu, Y., Zhu, K., & Liu, Z. (2021). Ssrnas: Search space reduced one-shot nas by a recursive attention-based predictor with cell tensor-flow diagram. In *2021 international joint conference on neural networks* (pp. 1–8). IEEE.

Lu, S., Hu, Y., Wang, P., Han, Y., Tan, J., Li, J., Yang, S., & Liu, J. (2023). Pinat: A permutation invariance augmented transformer for nas predictor. In *Proceedings of the AAAI conference on artificial intelligence*: *vol. 37*, (pp. 8957–8965).

Lu, S., Li, J., Tan, J., Yang, S., & Liu, J. (2021). TNASP: A transformer-based NAS predictor with a self-evolution framework. In *Advances in neural information processing systems*: *vol. 34*, (pp. 15125–15137).

Luo, Y., Thost, V., & Shi, L. (2023). Transformers over directed acyclic graphs. *Advances in Neural Information Processing Systems*, *36*, 47764–47782.

Ning, X., Zheng, Y., Zhou, Z., Zhao, T., Yang, H., & Wang, Y. (2023). A generic graph-based neural architecture encoding scheme with multifaceted information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*(7), 7955–7969.

Ning, X., Zhou, Z., Zhao, J., Zhao, T., Deng, Y., Tang, C., Liang, S., Yang, H., & Wang, Y. (2022). TA-GATES: An encoding scheme for neural network architectures. *Advances in Neural Information Processing Systems*, *35*, 32325–32339.

Peng, Y., Song, A., Fayek, H. M., Ciesielski, V., & Chang, X. (2024). SWAP-NAS: Sample-wise activation patterns for ultra-fast NAS. In *The twelfth international conference on learning representations*.

Poyser, M., & Breckon, T. P. (2024). Neural architecture search: A contemporary literature review for computer vision applications. *Pattern Recognition*, *147*, Article 110052.

Rampášek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., & Beaini, D. (2022). Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, *35*, 14501–14515.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, *115*(3), 211–252.

Safaei, B., Vibashan, V., De Melo, C. M., & Patel, V. M. (2024). Entropic open-set active learning. In *Proceedings of the AAAI conference on artificial intelligence*: *vol. 38*, (pp. 4686–4694).

Sener, O., & Savarese, S. (2018). Active learning for convolutional neural networks: A core-set approach. In *International conference on learning representations*.

Shi, H., Pi, R., Xu, H., Li, Z., Kwok, J., & Zhang, T. (2020). Bridging the gap between sample-based and one-shot neural architecture search with bonas. In *Advances in neural information processing systems*: *vol. 33*, (pp. 1808–1819).

Shu, Y., Dai, Z., Wu, Z., & Low, B. K. H. (2022). Unifying and boosting gradient-based training-free neural architecture search. In *Advances in neural information processing systems*: *vol. 35*, (pp. 33001–33015).

Sun, Y., Wang, H., Xue, B., Jin, Y., Yen, G. G., & Zhang, M. (2019). Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor. *IEEE Transactions on Evolutionary Computation*, *24*(2), 350–364.

Tan, W., Du, L., & Buntine, W. (2024). Bayesian estimate of mean proper scores for diversity-enhanced active learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *46*(5), 3463–3479.

Tang, Y., Wang, Y., Xu, Y., Chen, H., Shi, B., Xu, C., Xu, C., Tian, Q., & Xu, C. (2020). A semi-supervised assessor of neural architectures. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 1810–1819).

Thost, V., & Chen, J. (2021). Directed acyclic graph neural networks. In *International conference on learning representations*.

Wang, D., & Shang, Y. (2014). A new active labeling method for deep learning. In *2014 international joint conference on neural networks* (pp. 112–119). IEEE.

Wang, L., Zhao, Y., Jinnai, Y., Tian, Y., & Fonseca, R. (2019). Alphax: exploring neural architectures with deep neural networks and monte carlo tree search. arXiv preprint arXiv:1903.11059.

Wen, W., Liu, H., Chen, Y., Li, H., Bender, G., & Kindermans, P.-J. (2020). Neural predictor for neural architecture search. In *Computer vision–ECCV 2020: 16th European conference, Glasgow, UK, August 23–28, 2020, proceedings, part XXIX* (pp. 660–676). Springer.

White, C., Neiswanger, W., Nolen, S., & Savani, Y. (2020). A study on encodings for neural architecture search. In *Advances in neural information processing systems*: *vol. 33*, (pp. 20309–20319).

Wu, Z., Chen, J., Al-Sabri, R., Oloulade, B. M., & Gao, J. (2025). Asymmetric augmented paradigm-based graph neural architecture search. *Information Processing & Management*, *62*(1), Article 103897.

Xie, X., Sun, Y., Liu, Y., Zhang, M., & Tan, K. C. (2024). Architecture augmentation for performance predictor via graph isomorphism. *IEEE Transactions on Cybernetics*, *54*(3), 1828–1840.

Xu, Y., Wang, Y., Han, K., Tang, Y., Jui, S., Xu, C., & Xu, C. (2021). Renas: Relativistic evaluation of neural architecture search. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 4411–4420).

Xu, R., Zhang, H., Wang, Y., Zeng, W., & Zhang, S. (2025). NN-former: Rethinking graph structure in neural architecture representation. In *Proceedings of the computer vision and pattern recognition conference* (pp. 10004–10014).

Yi, Y., Zhang, H., Hu, W., Wang, N., & Wang, X. (2023). Nar-former: Neural architecture representation learning towards holistic attributes prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 7715–7724).

Ying, C., Klein, A., Christiansen, E., Real, E., Murphy, K., & Hutter, F. (2019). Nas-bench-101: Towards reproducible neural architecture search. In *International conference on machine learning* (pp. 7105–7114). PMLR.

Yu, C., Liu, X., Wang, Y., Liu, Y., Feng, W., Deng, X., Tang, C., & Lv, J. (2025). GPT-NAS: Neural architecture search meets generative pre-trained transformer model. *Big Data Mining and Analytics*, *8*(1), 45–64.

Zela, A., Siems, J. N., Zimmer, L., Lukasik, J., Keuper, M., & Hutter, F. (2022). Surrogate NAS benchmarks: Going beyond the limited search spaces of tabular NAS benchmarks. In *International conference on learning representations*.

Zhang, X., Ping, X., & Zhang, J. (2023). Deep active learning with structured neural depth search. arXiv preprint arXiv:2306.02808.

Zhao, X., Sun, J., Zhang, J., Hou, S., Li, S., Liu, T., & Liu, K. (2025). PerfSeer: An efficient and accurate deep learning models performance predictor. arXiv preprint arXiv:2502.01206.

Zhou, Q., Sheng, K., Zheng, X., Li, K., Sun, X., Tian, Y., Chen, J., & Ji, R. (2022). Training-free transformer architecture search. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10894–10903).

Zhou, D., Zhou, X., Zhang, W., Loy, C. C., Yi, S., Zhang, X., & Ouyang, W. (2020). Econas: Finding proxies for economical neural architecture search. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11396–11404).