ACDP: Architecture-aware Cross-Dataset Performance Predictor for NAS

Jiawen Deng Yuqi Feng Han Ji Yanan Sun*
College of Computer Science, Sichuan University, China
dengjiamin@stu.scu.edu.cn, feng770623@gmail.com
jihan.ji@connect.polyu.hk, ysun@scu.edu.cn
*Corresponding author

Abstract

Performance predictors are pivotal in accelerating evaluation in NAS. Cross-dataset predictors are trained on (dataset, architecture, performance) triplets to efficiently estimate architectural performance for unseen datasets. However, existing cross-dataset predictors rely on a naive concatenation of dataset and architecture features. This fusion ignores the architecture-aware nature, where the same dataset manifests as distinct representations under different architectures, thereby limiting generalization. To tackle this issue, we propose ACDP, an Architecture-aware Cross-Dataset Performance Predictor that dynamically adjusts dataset feature extraction based on specific architecture. Initially, ACDP develops an elastic mechanism to extract raw dataset traits, providing a configurable lever for the trade-off between efficiency and accuracy. Furthermore, ACDP leverages a hypernetwork to determine dataset feature extractor parameters conditioned on architectural features, transforming raw dataset traits into tailored representations. ACDP excels in both specialization and generalization on six unseen datasets. Notably, in NAS-Bench-201, ACDP achieves 0.759/0.721 Kendall's Tau on CIFAR-10/100, surpassing single-dataset SOTAs trained with over 100 target samples. In MobileNetV3, ACDP discovers architectures outperforming cross-dataset SOTAs, notably gaining +2.04% accuracy on Aircraft. Code: `https://github.com/Thorn-222/ACDP`.

1 Introduction

Neural Architecture Search (NAS) automates the design of neural network architectures, where performance evaluation is the critical step. However, performing full training for each architecture leads to immense computational and time costs [15]. To mitigate this bottleneck, the performance predictor is introduced to efficiently predict performance, replacing the full training process [42].

Most existing performance predictors adhere to the "single-dataset paradigm" that restricts their utility to the original training dataset, thereby rendering them inapplicable to unseen datasets [36]. To bypass the time-consuming retraining required by this paradigm, emerging cross-dataset predictors [25] expand the training space from (architecture, performance) pairs into (dataset, architecture, performance) triplets. By incorporating dataset-level features, these predictors aim to abstract evaluation rules across heterogeneous datasets.

Emerging cross-dataset predictors rely on naive feature concatenation of dataset and architecture as input to evaluate performance [25, 2]. However, such static feature fusion treats datasets and architectures as independent entities, failing to model the intricate coupling between them and thus leading to poor generalization. Specifically, this coupling is manifested by the fact that the effective features of a dataset vary according to the properties of the architecture. For instance, when facing the same dataset, a complex architecture may focus on global patterns, whereas a lightweight one might perceive local statistics, leading to disparate features and resultant performance [49]. Such ar-

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

chitecture-aware behavior suggests that dataset features are not fixed but rather a dynamic product of the interaction with the architecture. Consequently, static fusion fails to capture these essential interdependent dynamics by indiscriminately applying the same dataset representation across all architectures. To bridge this gap, we argue that dataset features must be adaptively extracted in a way that is conditioned on the architecture being evaluated.

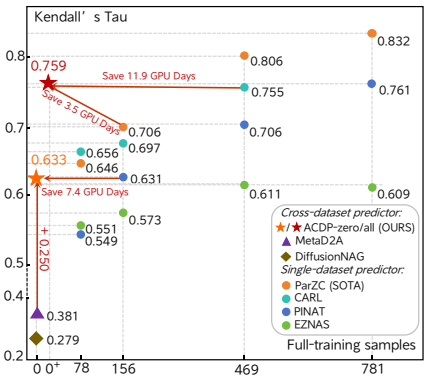

Figure 1: Kendall's Tau vs. Full-training samples on CIFAR-10 in NAS-Bench-201.

Driven by this crucial insight, we propose an Architecture-aware Cross-Dataset Performance Predictor (ACDP) in this work, which implements architecture-aware dataset features to enhance cross-dataset generalization. Specifically, to strike a practical balance between efficiency and accuracy, we first design an elastic mechanism to capture raw dataset traits based on computational budgets. Users can either exclusively employ zero-cost intrinsic expressivity metrics as the raw dataset traits to maximize efficiency or incorporate low-fidelity probing (e.g., accuracy after 5 training epochs) to reinforce these traits for enhanced precision. Furthermore, to achieve architecture awareness, we design a hypernetwork-driven dataset feature extractor. This hypernetwork is conditioned on architectural features to dynamically modulate the parameters of the dataset feature extractor. Through this extractor, the raw dataset traits are adaptively projected into a space tailored to each specific architecture. These architecture-aware dataset features are subsequently mapped to the final performance scores. In practical search scenarios on unseen datasets, ACDP provides a hierarchical strategy: it first utilizes zero-cost dataset traits for architecture-aware prediction to rapidly prune a massive number of candidates, followed by low-fidelity probing to achieve refined re-ranking for a few high-potential architectures. Notably, the hierarchical ratio, which determines the percentage of candidates receiving probing, can be flexibly adjusted. The primary contributions of this work are as follows:

- We design an elastic mechanism to extract raw dataset traits. It can either utilize zero-cost intrinsic expressivity metrics for expedited evaluation or incorporate low-fidelity probing to reinforce these traits. Such flexibility provides a unified framework for hierarchical search, improving the trade-off between efficiency and accuracy in real-world NAS applications.
- We introduce an architecture-aware dataset feature extractor. By dynamically modulating parameters, it tailors dataset features for specific architecture, overcoming the limitation of static extractors and enhancing generalization for unseen dataset-architecture combinations.
- We demonstrate ACDP's superior generalization across six datasets under various hierarchical ratios. In NAS-Bench-201, in terms of Kendall's Tau on CIFAR-10 (Fig. 1), zero-cost ACDP significantly outperforms cross-dataset SOTA (0.633 vs. 0.381), and 100%-probing ACDP surpasses single-dataset SOTA utilizing hundreds of full-training samples (0.759 vs. 0.706), while saving 3.5 GPU days. In MobileNetV3 space, 10%-probing ACDP discovers architectures that notably gain +2.04% accuracy on Aircraft over cross-dataset SOTA.

2 Related Work

2.1 Cross-Dataset NAS Performance Predictors

We categorize current cross-dataset predictors into two major directions. The first is transfer learning, which adapts source-trained predictors to target datasets through fine-tuning, such as MP-NAS [29] and TRP [28]. However, these methods inherently rely on target-domain ground-truth samples for calibration, hindering true generalization. The second is meta-learning, which uses (dataset, architecture, performance) triplets to achieve a "train once, apply universally" goal. We hereafter reserve "cross-dataset" exclusively for this paradigm.

Notably, MetaD2A [25] pioneered this "cross-dataset" path by constructing the first large-scale triplet database and proposing a meta-predictor. DiffusionNAG [2] utilized this dataset to train a new cross-

dataset predictor. Critically, both methods primarily focus on the generation of architectures. Their predictors are treated as auxiliary components and consistently employ feature fusion through simple concatenation, which fails to capture deep coupling between datasets and architectures. Unlike these works, we focus on the fundamental modeling of the cross-dataset predictor itself. By proposing an architecture-aware mechanism, ACDP attains superior generalization with adaptive characterization.

2.2 Hypernetworks

Hypernetworks provide a powerful mechanism for dynamic parameter generation, using one network to generate weights for another [17]. By learning a mapping from task-specific inputs to a high-dimensional parameter space, they overcome the limitations of fixed-weight models and enable input-dependent dynamic modeling. In the NAS field, hypernetworks have evolved across three key stages: SMASH [4] first used simple bit-mask encodings to generate architecture weights; GHN [48] employed GNNs to model complex graph topologies; and recently, HyperNAS [33] integrated these techniques into modern weight-sharing frameworks to maximize search efficiency.

Inspired by these advances, we integrate a hypernetwork into the cross-dataset predictor framework. By dynamically generating parameters for the dataset feature extractor based on architectural features, ACDP effectively captures the intricate coupling between architectures and datasets.

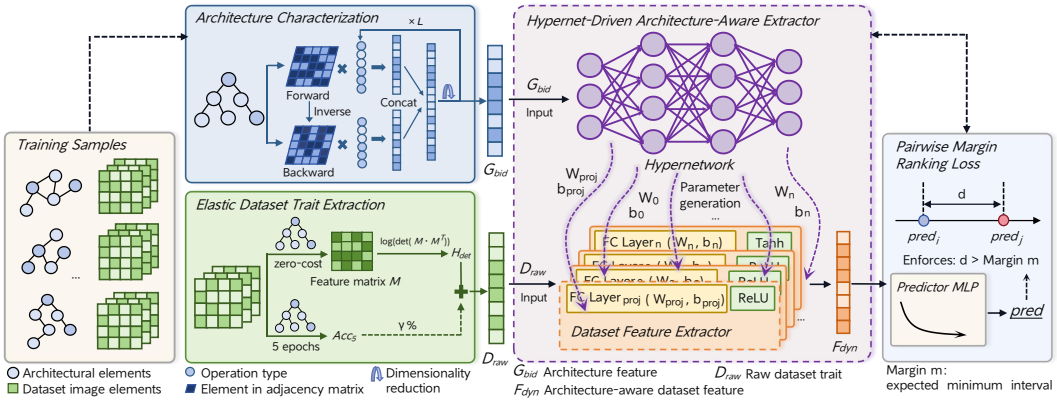

Figure 2: Overview of ACDP: Training samples comprise architectures and datasets. The architectural feature G_{bid} is input to the hypernetwork to generate parameters for the dataset feature extractor, which transforms the dataset trait D_{raw} to the architecture-aware dataset feature F_{dyn}.

3 Method

As shown in Figure 2, ACDP consists of training samples, architecture characterization, dataset potential trait, hypernet-driven architecture-aware extractor, and pairwise margin ranking loss.

3.1 Training Samples

ACDP utilizes a sample set $T = \{(d_i, a_i, y_i)\}_{i=1}^{t}$ for training, where t is the total number of triplets. Each triplet consists of a dataset d_i sampled from the dataset space D, an architecture a_i from the search space A, and y_i is the ground-truth accuracy of a_i on dataset d_i.

Each architecture is represented as a graph $\mathcal{G} = (N, O)$. For architectures falling into Directed Acyclic Graph (DAG)-based space NAS-Bench-201 [13], the adjacency matrix N captures the topology, while the operation matrix O encodes node operations. For those in chain-structured space MobileNetV3 [19], N defines the sequential connections and O represents block-wise attributes, including kernel size, expansion, and depth.

3.2 Architecture Characterization

To encode architectural topology, we follow the best practice in the NAS community, using a Bidirectional Graph Convolutional Network (Bi-GCN) [44]. This captures bidirectional data flow by

processing the adjacency matrix N and its transpose $N^\top$ in parallel. Following is the calculation for the bidirectional features, $X_{\text{f}}^{(l)}$ and $X_b^{(l)}$:

$$X_f^{(l)} = \text{ReLU}(\tilde{N} X_f^{(l-1)} W_f^{(l)}) \tag{1}$$

$$X_b^{(l)} = \text{ReLU}(\tilde{N}^\top X_b^{(l-1)} W_b^{(l)}), \tag{2}$$

where $l \in \{1, \ldots, L\}$ denotes the layer index, and $X_f^{(0)}$ and $X_b^{(0)}$ are set to the operation matrix O defined in Section 3.1. Here, $\tilde{N}$ represents the normalized adjacency matrix with self-loops, and $W^{(l)}$ are trainable weights for each stream. The final architectural feature G_{bid} is derived by fusing the dual-stream outputs from the last layer:

$$G_{bid} = \text{MeanPool}(\text{Linear}(\text{Concat}(X_f^{(L)}, X_b^{(L)}))). \tag{3}$$

This representation encapsulates the comprehensive structural context, serving as a global descriptor to guide the subsequent dataset feature extraction.

3.3 Elastic Dataset Trait Extraction

To ensure that ACDP is not only specialized for computer vision but also seamlessly extensible to diverse task domains, we propose utilizing architecture-to-dataset response indicators as the raw dataset traits. This design avoids reliance on domain-specific trait extractors (e.g., pre-trained CNNs for images). Specifically, based on computational budget, we provide two elastic trait configurations:

Zero-cost Configuration: Intrinsic Expressivity (H_{det}). As the most fundamental signal, we utilize a training-free geometric metric to capture structural priors, defined as:

$$H_{det} = \log(|\det(M)|). \tag{4}$$

M is the activation overlap matrix from a single forward pass with zero cost. This quantifies the diversity of trait representations by measuring the volume spanned by activation vectors [35]. A larger H_{det} signifies a better capacity to capture diverse and independent information from input data.

Reinforced Configuration: Low-fidelity Probing (Acc_5). While H_{det} reflects the geometric capacity of the feature space as a theoretical upper bound, it occasionally overlooks the non-linear complexities inherent in actual training. To achieve higher precision, we optionally incorporate low-fidelity probing, specifically the accuracy obtained after 5 training epochs (Acc_5). By fusing these two indicators, we derive a "hybrid-fidelity" trait, thereby bridging the gap between theoretical potential and empirical realization.

Elastic Integration. The raw dataset trait D_{raw} is no longer a static entity but a flexible, configurable lever. Users can seamlessly switch configurations based on their specific requirements:

$$D_{raw} = \begin{cases} \text{Norm}(H_{det}) & \text{high-efficiency mode} \\ \text{Norm}(H_{det}) + \text{Norm}(Acc_5) & \text{high-accuracy mode} \end{cases} \tag{5}$$

where $\text{Norm}(x) = (x - \min)/(\max - \min)$. The boundary statistics $\{\min, \max\}$ are computed based on the set of candidate architectures to be predicted for the current mode.

3.4 Hypernet-Driven Architecture-Aware Extractor

ACDP employs a hypernetwork $\mathcal{H}_{\text{Dyn}}$ to dynamically generate the parameters of the extractor $\mathcal{M}$ by architecture feature G_{bid}, yielding architecture-aware dataset features F_{dyn}.

3.4.1 Hypernet-driven weight generation

To efficiently model the hypernet-driven architecture-aware mechanism, $\mathcal{H}_{\text{Dyn}}$ and $\mathcal{M}$ are built as h and n layers of Multi-Layer Perceptrons (MLPs). By the structural richness of G_{bid} and discriminative priors of D_{raw}, these low-complexity mappers attain superior efficiency and efficacy.

Given potential dimension mismatches between the raw input $D_{raw} \in \mathbb{R}^{d_{\text{raw}}}$ and the architecture-aware feature $F_{dyn} \in \mathbb{R}^{d_{\text{dyn}}}$, $\mathcal{H}_{\text{Dyn}}$ adaptively generates an initial projection layer L_{proj} to unify the

feature dimensions when necessary. Accordingly, the parameter index set K is defined as:

$$K = \begin{cases} \{\text{proj}, 1, 2, \ldots, n\} & \text{if } d_{\text{raw}} \neq d_{\text{dyn}} \\ \{1, 2, \ldots, n\} & \text{if } d_{\text{raw}} = d_{\text{dyn}}\,. \end{cases} \tag{6}$$

Subsequently, the architecture feature G_{bid} serves as the input to the hypernetwork $\mathcal{H}_{\text{Dyn}}$. $\mathcal{H}_{\text{Dyn}}$ produces a unified parameter pool Θ_{pool} in the form of a high-dimensional vector:

$$\Theta_{\text{pool}} = \mathcal{H}_{\text{Dyn}}(G_{bid}; V_{\mathcal{H}}), \tag{7}$$

where $V_{\mathcal{H}}$ denotes the set of all trainable parameters within the $\mathcal{H}_{\text{Dyn}}$. The parameter set $\Theta_{\mathcal{M}}$ for $\mathcal{M}$ is then obtained by segmenting and reshaping the pool:

$$\Theta_{\mathcal{M}} = \{\Phi_k \mid \Phi_k = \text{Segment}(\Theta_{\text{pool}}, k), \forall k \in K\}. \tag{8}$$

$\Phi_k = (W_k, b_k)$ represents the weights and biases for the k-th layer. $\text{Segment}(\cdot, k)$ extracts and reshapes the corresponding segments from Θ_{pool} according to the dimensions of each layer in $\mathcal{M}$.

3.4.2 Architecture-aware feature extraction

After updating parameters, $\mathcal{M}$ receives D_{raw} as input and outputs architecture-aware features F_{dyn}:

$$F_{dyn} = \mathcal{M}(D_{raw}; \Theta_{\mathcal{M}}). \tag{9}$$

The computation of F_{dyn} can be recursively represented as:

$$\begin{aligned} f_0 &= \begin{cases} D_{raw} & \text{if } d_{\text{raw}} = d_{\text{dyn}} \\ D_{raw} W_{\text{proj}} + b_{\text{proj}} & \text{if } d_{\text{raw}} \neq d_{\text{dyn}}, \end{cases} \\ f_i &= \text{ReLU}(f_{i-1} W_i + b_i), \quad i = 1, \ldots, n-1. \end{aligned} \tag{10}$$

$$F_{dyn} = \text{Tanh}(f_{n-1} W_n + b_n). \tag{11}$$

Notably, although D_{raw} is low-dimensional, it serves as a latent anchor to trigger the mapping. This "small input, large parameters" design avoids the informational redundancy associated with high-dimensional features while ensuring to capture complex non-linear relationships.

This mechanism enables a deep modulation of dataset traits D_{raw} conditioned on architectural features G_{bid}. Consequently, the resulting F_{dyn} transcends basic dataset statistical descriptors and captures the high-order interactions between dataset traits and specific architectural structures, thereby enhancing the generalization across diverse datasets.

3.5 Pairwise Margin Ranking Loss

With $\hat{y}$ mapped from F_{dyn} via an MLP, we first consider a pairwise ranking loss to optimize ACDP:

$$\mathcal{L}_{\text{classic}} = \frac{1}{|E|} \sum_{(a_i, a_j) \in E} \log\left(1 + \exp\left(-\sigma_{ij} d_{ij}\right)\right). \tag{12}$$

y_i is the true accuracy of a_i. $d_{ij} = \hat{y}_i - \hat{y}_j$, $\sigma_{ij} = \text{sgn}(y_i - y_j)$. E is the set of architecture pairs.

However, the gradient vanishes when predicted scores are close, leading to insufficient penalty. To formally optimize ACDP with enhanced sorting resolution, we introduce an indicator function $\mathbb{I}_{y_i \neq y_j}$ to exclude accuracy ties and a fixed margin m to enforce stricter separation ($\sigma_{ij} d_{ij} > m$):

$$\mathcal{L} = \frac{1}{|E|} \sum_{(a_i, a_j) \in E} \mathbb{I}_{y_i \neq y_j} \cdot \log\left(1 + \exp\left(m - \sigma_{ij} d_{ij}\right)\right). \tag{13}$$

3.6 Hierarchical Evaluation Strategy

To push the trade-off between efficiency and accuracy to its limit, we propose an efficient hierarchical evaluation strategy in practical search. Prior to deployment, we define two variants, the high-efficiency ACDP-zero (optimized via H_{det}) and the high-accuracy ACDP-all (optimized via $H_{det} + Acc_5$). When navigating an entirely new dataset, ACDP executes a two-stage evaluation pipeline:

Table 1: Comparison of Kendall's Tau on CIFAR-10.

Method	Train Samples	Test Samples	Kendall's τ
NAO [32]	781 (5%)	all	0.522
NP [44]	781 (5%)	all	0.634
EZNAS [1]	781 (5%)	all	0.609
TNASP [31]	469 (3%)	all	0.640
PINAT [30]	469 (3%)	all	0.706
CAP [21]	469 (3%)	all	0.755
CARL [20]	469 (3%)	all	0.755
ParZC [10]	156 (1%)	all	0.706
MetaD2A [25]	0	all	0.381
DiffusionNAG [2]	0	all	0.280
ACDP-zero (Ours)	**0**	**all**	**0.633**
ACDP-all (Ours)	**5-epoch for all**[†]	**all**	**0.759**

[†]: ACDP-all uses 5-epoch probing for full samples.

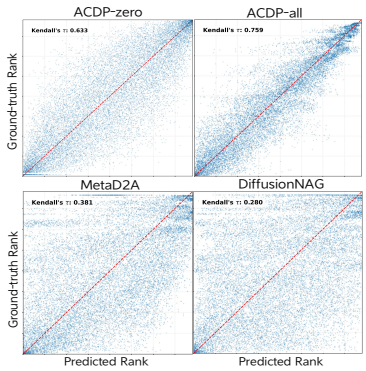

Figure 3: Rank correlation.

Stage 1: Zero-cost Rapid Pruning. For the massive initial search space, we first employ ACDP-zero for global performance estimation. Acting as a "high-pass filter," it rapidly prunes a vast majority of low-potential architectures at near-zero cost, significantly narrowing the candidate pool.

Stage 2: Low-fidelity Refined Re-ranking. For the top-γ (e.g., 10%) high-potential candidates retained from the first stage, we then allocate computational budget to obtain their Acc_5. These reinforced traits are fed into ACDP-all for a precise re-ranking to identify the optimal architectures.

Through this hierarchical strategy, ACDP ensures that expensive computational probing is strictly reserved for a small fraction of promising candidates. The hierarchical ratio γ serves as a configurable lever, granting users the flexibility to adjust resource allocation according to the specific requirements.

4 Experiments

In this section, we evaluate ACDP under various hierarchical ratios on two distinct search spaces, including NAS-Bench-201 [13] and MobileNetV3 [19], and assess its performance across six datasets: CIFAR-10, CIFAR-100 [24], MNIST [9], SVHN [37], Aircraft [34], and Pets [38].

4.1 Experimental Settings

ACDP is trained on the triple set $T = \{(d_i, a_i, y_i)\}_{i=1}^{t}$ provided by MetaD2A [25]. $d_i \in D$ denotes a dataset, $a_i \in A$ represents an architecture sampled from the search space, and y_i is the corresponding ground-truth performance. Specifically, the dataset collection D is constructed by sampling class-specific subsets from ImageNet-1K [8]. For NAS-Bench-201, each dataset in D consists of 20 classes, providing $t = 4,630$ triplet samples in total. For the more complex MobileNetV3 space, the class counts per dataset randomly span from 50 to 1,000, resulting in $t = 4,310$ samples.

The architecture encoder consists of a 6-layer Bi-GCN. The $\mathcal{H}_{\text{Dyn}}$ and the $\mathcal{M}$ are implemented as 2-layer and 3-layer MLPs. The margin m in the pairwise margin ranking loss is fixed at 0.5.

4.2 Experimental Results and Analysis

We evaluate ACDP on NAS-Bench-201 across two dimensions of effectiveness and practical utility. Subsequently, evolutionary search experiments in the complex MobileNetV3 space further demonstrate its scalability and superiority in large-scale search scenarios.

4.2.1 Results in NAS-Bench-201

Effectiveness Analysis Following the protocol in TNASP [31], we benchmark ACDP on CIFAR-10 using Kendall's Tau (τ). As demonstrated in Table 1, without accessing any training samples from the target dataset, ACDP-zero achieves a τ of 0.633, significantly outperforming zero-cost cross-dataset baselines such as MetaD2A and DiffusionNAG. This underscores the reliable discriminative power of our architecture-aware mechanism even with minimal information input. Upon incorporating Acc_5 signals, the precision of ACDP-all substantially increases to 0.759, which not only far exceeds all cr-

Table 2: Practical utility across six datasets in Nas-Bench-201. (P: Parameters, A: Accuracy)

Method	CIFAR-10		CIFAR-100		MNIST		SVHN		Aircraft		Pets	
	P (M)	A (%)	P (M)	A (%)	P (M)	A (%)	P (M)	A (%)	P (M)	A (%)	P (M)	A (%)
ResNet [18]	0.86	93.97±0.00	0.86	70.86±0.00	0.86	99.67±0.01	0.86	96.13±0.19	0.86	47.01±1.16	0.86	25.58±3.43
RSPS [27]	–	84.07±3.61	–	52.31±5.77	0.25	99.63±0.02	0.48	96.17±0.12	0.22	42.19±3.88	0.32	22.91±1.65
SETN [11]	–	87.64±0.00	–	59.09±0.24	0.56	99.69±0.04	0.48	96.02±0.12	0.44	44.84±3.96	0.32	25.17±1.68
GDAS [12]	–	93.61±0.09	–	70.70±0.30	0.82	99.64±0.04	0.24	95.57±0.57	0.62	53.52±0.48	0.83	24.02±2.75
PC-DARTS [46]	1.17	93.66±0.17	0.26	66.64±2.34	0.62	99.66±0.04	0.47	95.40±0.67	0.32	26.33±3.40	0.44	25.31±1.38
DrNAS [6]	1.53	94.36±0.00	1.20	**73.51±0.00**	1.53	99.59±0.04	1.53	96.30±0.05	1.03	46.08±7.00	0.44	26.73±2.61
TNAS [41]	1.04	94.16±0.11	0.85	73.02±0.12	0.13	99.73±0.03	0.85	96.54±0.05	1.05	56.24±0.55	0.85	39.56±0.81
MetaD2A [25]	1.08	**94.37±0.00**	1.04	**73.51±0.00**	1.05	99.69±0.10	1.05	96.51±0.08	1.05	56.81±0.31	1.05	41.14±2.30
DiffusionNAG [2]	1.28	**94.37±0.00**	1.28	**73.51±0.00**	1.28	99.73±0.08	1.28	96.42±0.16	1.28	56.93±0.31	1.28	38.80±7.14
ACDP-zero (Ours)	0.80	**94.37±0.00**	0.85	**73.51±0.00**	0.80	99.73±0.03	0.83	96.52±0.01	0.80	57.22±0.22	0.83	41.22±0.08
ACDP-10% (Ours)	0.80	**94.37±0.00**	1.04	**73.51±0.00**	0.80	**99.74±0.02**	0.80	**96.57±0.01**	0.80	**57.35±0.28**	0.83	**41.26±0.18**

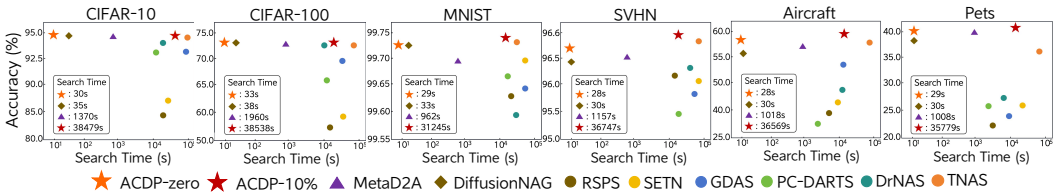

Figure 4: Search time across six datasets in Nas-Bench-201.

oss-dataset SOTAs but also surpasses single-dataset SOTAs that require hundreds of full-training samples. Notably, even in the "full-probing" configuration, ACDP-all still saves approximately 3.5 GPU days compared to single-dataset predictors that necessitate retraining from scratch, and the detailed calculations are provided in Appendix C.1.

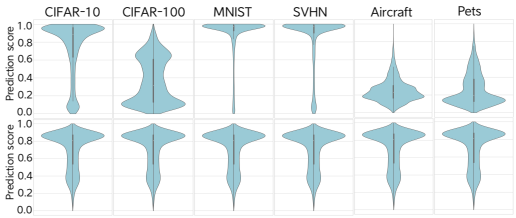

Figure 5: ACDP-zero (top) vs. MetaD2A (bottom). Predicted score and density of architectures.

To further investigate the behavior of the predictors, we mapped predicted versus actual ranks for all architectures. As shown in Figure 3, both ACDP-zero and ACDP-all exhibit a strong monotonic alignment with the diagonal. Particularly in the critical top-performance regime (bottom-left) and the low-performance regime (top-right), this tight alignment ensures that we can identify high-quality architectures while rapidly pruning inferior candidates. In contrast, MetaD2A and DiffusionNAG display severe global dispersion and almost entirely lose discriminative granularity. This visual evidence further confirms the better predictive ability of ACDP in cross-dataset scenarios.

Practical Utility We evaluate ACDP's practical utility in discovering optimal architectures. To directly compare with baselines, we conduct a global search across the entire NAS-Bench-201 space on six datasets (CIFAR-10/100, MNIST, SVHN, Aircraft, and Pets). We report results for both the ACDP-zero and the hierarchical ACDP-10%. For the latter, ACDP-zero first filters out the bottom 90% of architectures, and ACDP-all performs refined prediction on the top 10% of candidates.

As shown in Table 2 and Figure 4, zero-cost cross-dataset predictors significantly reduce search time compared to traditional NAS methods by obviating the need for costly full training of candidates on target datasets. Among them, ACDP-zero achieves substantial improvements over existing cross-dataset SOTAs with minimum search time. By performing more granular probing on top-tier candidates, ACDP-10% further elevates predictive precision. While MetaD2A and DiffusionNAG can also identify competitive architectures, their search results consistently converge toward high-parameter structures, often selecting identical architectures even across disparate datasets. This suggests that they may rely on a "higher parameters equal higher performance" bias.

Table 3: Evolutionary search across six datasets in MobileNetV3. (T: Search Time, A: Accuracy)

Dataset	**MetaD2A**		**DiffusionNAG**		**ACDP-zero**		**ACDP-10%**	
	T (s)	A (%)	T (s)	A (%)	T (s)	A (%)	T (s)	A (%)
CIFAR-10	71	97.70±0.11	54	97.88±0.11	**51**	97.92±0.06	22k	**98.04±0.04**
CIFAR-100	82	85.57±0.21	61	85.93±0.18	**55**	86.22±0.15	24k	**87.45±0.12**
MNIST	68	99.54±0.05	41	99.56±0.04	**42**	99.63±0.04	8.2k	**99.66±0.08**
SVHN	73	96.74±0.08	50	96.82±0.10	**49**	96.98±0.14	26k	**97.11±0.12**
Aircraft	89	77.47±0.35	66	77.53±0.41	**61**	78.68±0.07	44k	**79.57±0.02**
Pets	62	93.96±0.15	54	94.14±0.22	**56**	94.51±0.13	24k	**94.77±0.11**

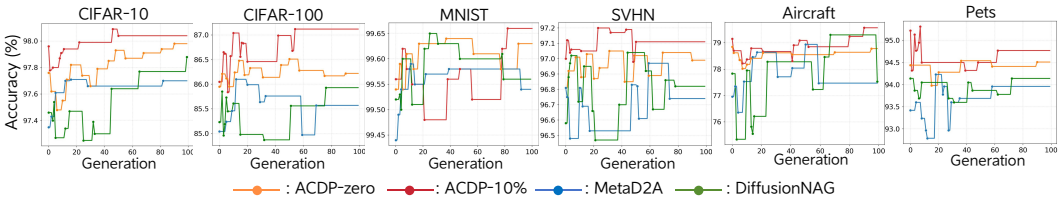

Figure 6: Evolutionary search trajectories across six datasets in MobileNetV3.

To further elucidate this, we visualize the distribution density of predicted scores for all architectures in the search space across six datasets (Figure 5). The cause of MetaD2A's (bottom) bias becomes evident: its predicted profiles remain largely invariant regardless of the target dataset's complexity, exposing a critical deficiency in cross-dataset awareness. In contrast, without accessing any training label or 5-epoch probing from the target dataset, ACDP-zero (top) exhibits a "Distribution Shift" that aligns with dataset difficulty. For simpler datasets (e.g., CIFAR-10, MNIST and SVHN), predictions are highly skewed toward the upper bound. Conversely, for challenging datasets (e.g., CIFAR-100, Aircraft and Pets), the distribution centers shift significantly toward the lower-middle range, with a marked reduction in architectures that are predicted to have high accuracy. This sensitivity confirms that ACDP with architecture-aware mechanisms can distill dataset-specific semantics to modulate its evaluative logic to match distinct, high-performing architectures to specific datasets.

4.2.2 Results in MobileNetV3

To verify the reliability of ACDP in ultra-large and non-enumerable spaces, we extend our evaluation to the MobileNetV3 space (10^{19} candidates). We adopt an evolutionary search strategy where ACDP-zero, ACDP-10%, MetaD2A, and DiffusionNAG serve as the evaluation engines for their respective search processes. Starting from an identical initial population of 30 candidates, each predictor identifies the top 10 candidates per generation for mutation, iterating for 100 generations. For ACDP-10%, we implement a hierarchical search strategy on each new population to enhance accuracy. We execute this search pipeline on six datasets to ensure a comprehensive evaluation. Finally, the optimal architectures discovered by each method are trained from scratch to obtain their actual accuracies, validating the decision-making utility of the predictors in a real-world environment.

As shown in Table 3, ACDP consistently identifies superior architectures across all six datasets under both configurations. The search trajectories (Figure 6) further reveal the distinct search dynamics of each predictor. Specifically, while all methods start from identical initial populations, ACDP immediately identifies high-performance candidates in the first iteration. ACDP-zero maintains a steady performance profile that consistently exceeds cross-dataset SOTAs in minimum search time. ACDP-10% further elevates the precision, such as on the challenging dataset Aircraft, yielding a substantial gain of $+2.04\%$ over the strongest baseline. In contrast, the trajectories of MetaD2A and DiffusionNAG fluctuate violently, where "optimal" candidates identified often yield lower actual performance than their predecessors, exposing a critical lack of discriminative signals in unseen datasets. These results underscore that ACDP's architecture-aware logic enables efficient navigation, while its hierarchical strategy provides an additional performance boost in ultra-large search spaces.

4.3 Ablation Study

We perform extensive ablation studies of ACDP from quantitative and qualitative perspectives on CIFAR-10/100 in NAS-Bench-201, validating the numerical effectiveness and representation logic.

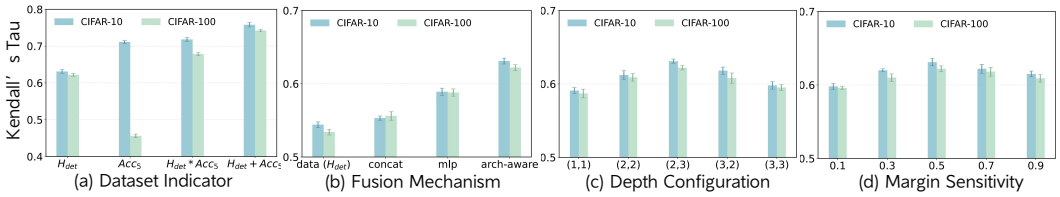

Figure 7: Quantitative assessment of design on CIFAR-10/100 in NAS-Bench-201.

4.3.1 Quantitative Performance Ablation

Dataset Indicator As shown in Figure 7 (a), using only H_{det}, the model achieves a reliable performance. When Acc_5 is introduced, the precision improves significantly. This demonstrates the utility of the hierarchical strategy. We also conduct comprehensive ablations on Acc_5 and hybrid indicators in Appendix C.5. Henceforth, ablation experiments exclusively utilize ACDP-zero.

Fusion Mechanism As shown in Figure 7 (b), the gains brought by MLP mapping of H_{det} or simple concatenation with architecture features are minimal. Conversely, our architecture-aware module yields a significant leap, proving the distilling of task-specific features.

Depth Configuration As shown in Figure 7 (c), we explore depth pairs for the hypernetwork and architecture-aware extractor. The (2,3) configuration emerges as the "sweet spot."

Margin Sensitivity As shown in Figure 7 (d), either too small or too large a margin hinders optimization convergence, while a moderate value of $m = 0.5$ yields the best performance.

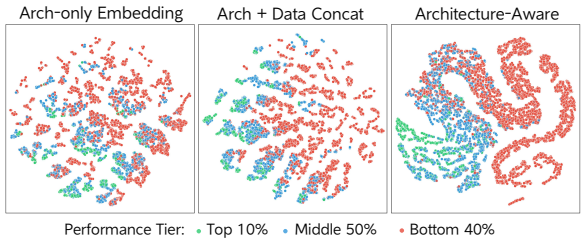

Figure 8: Progressive t-SNE visualization of representation space on CIFAR-10 in NAS-Bench-201.

4.3.2 Qualitative Representation Visualization

From a qualitative perspective, we investigate the role of the architecture-aware mechanism in reshaping the representation space. Specifically, we extract features from the predictor's final layer and project them onto a 2D plane via t-SNE [43], where each architecture is color-coded by its ground-truth accuracy tier. We use the ACDP-zero to isolate the influence of target datasets probing. Figure 8 demonstrates the evolution of the representation space from "disordered" to "aware."

Initially, using only architectural features exhibits a highly disordered distribution. Subsequently, concatenating architectural features with the dataset traits H_{det} introduces a preliminary linear stratification. This demonstrates that H_{det} effectively alleviates ambiguity of architectural features. However, significant overlap still remains between performance boundaries due to the lack of deep fusion. Finally, with architecture-aware embeddings, the feature space undergoes a non-linear reconstruction, evolving into distinct "island-like" clusters. This qualitative leap from chaotic entanglement to linear stratification and ultimately to manifold clustering validates that the architecture-aware module successfully captures the complex coupling between architectures and datasets.

5 Conclusion

This paper presents ACDP, a performance predictor achieving superior cross-dataset generalization in NAS. By employing a hypernetwork to dynamically modulate dataset feature extraction based on architectures, ACDP transforms raw traits into tailored representations for effective cross-dataset evaluation. To balance efficiency and precision in practical search, we also introduce a hierarchical search strategy. ACDP significantly outperforms single/cross-dataset baselines in diverse datasets.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China under Grant 62276175 and the Sichuan Science and Technology Program under Grant 2026JDDQ0004. The authors declare no competing interests.

References

[1] Yash Akhauri, Juan Munoz, Nilesh Jain, and Ravishankar Iyer. Eznas: evolving zero-cost proxies for neural architecture scoring. *Advances in Neural Information Processing Systems*, 35:30459–30470, 2022.

[2] Sohyun An, Hayeon Lee, Jaehyeong Jo, Seanie Lee, and Sung Ju Hwang. Diffusionnag: Predictor-guided neural architecture generation with diffusion models. In *12th International Conference on Learning Representations, ICLR 2024*, 2024.

[3] Thomas Bartz-Beielstein, Jürgen Branke, Jörn Mehnen, and Olaf Mersmann. Evolutionary algorithms. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(3):178–195, 2014.

[4] Andrew Brock, Theodore Lim, JM Ritchie, and Nick Weston. Smash: One-shot model architecture search through hypernetworks. 2018.

[5] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

[6] Xiangning Chen, Ruochen Wang, Minhao Cheng, Xiaocheng Tang, and Cho-Jui Hsieh. Drnas: Dirichlet neural architecture search. *arXiv preprint arXiv:2006.10355*, 2020.

[7] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

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

[9] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

[10] Peijie Dong, Lujun Li, Zhenheng Tang, Xiang Liu, Zimian Wei, Qiang Wang, and Xiaowen Chu. Parzc: Parametric zero-cost proxies for efficient nas. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 16327–16335, 2025.

[11] Xuanyi Dong and Yi Yang. One-shot neural architecture search via self-evaluated template network. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3681–3690, 2019.

[12] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1761–1770, 2019.

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

[14] Lukasz Dudziak, Thomas Chau, Mohamed Abdelfattah, Royson Lee, Hyeji Kim, and Nicholas Lane. Brp-nas: Prediction-based nas using gcns. *Advances in neural information processing systems*, 33:10480–10490, 2020.

[15] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

[16] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. Graph neural architecture search. In *International joint conference on artificial intelligence*. International Joint Conference on Artificial Intelligence, 2021.

[17] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[19] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1314–1324, 2019.

[20] Han Ji, Yuqi Feng, Jiahao Fan, and Yanan Sun. Carl: Causality-guided architecture representation learning for an interpretable performance predictor. *arXiv preprint arXiv:2506.04001*, 2025.

[21] Han Ji, Yuqi Feng, and Yanan Sun. Cap: a context-aware neural predictor for nas. *arXiv preprint arXiv:2406.02056*, 2024.

[22] Tangyu Jiang, Haodi Wang, and Rongfang Bie. Meco: zero-shot nas with one data and single forward pass via minimum eigenvalue of correlation. *Advances in Neural Information Processing Systems*, 36:61020–61047, 2023.

[23] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[24] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[25] Hayeon Lee, Eunyoung Hyung, and Sung Ju Hwang. Rapid neural architecture search by learning to generate graphs from datasets. In *9th International Conference on Learning Representations, ICLR 2021*, 2021.

[26] Guihong Li, Yuedong Yang, Kartikeya Bhardwaj, and Radu Marculescu. Zico: Zero-shot nas via inverse coefficient of variation on gradients. *arXiv preprint arXiv:2301.11300*, 2023.

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

[28] Nan Li, Bing Xue, Lianbo Ma, and Mengjie Zhang. Transferable relativistic predictor: Mitigating cross-task cold-start issue in nas. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, pages 5625–5633, 2025.

[29] Hrushikesh Loya, Abhinav Mehrotra, Royson Lee, Javier Fernandez-Marques, Nicholas D Lane, Hongkai Wen, et al. How much is hidden in the nas benchmarks? few-shot adaptation of a nas predictor. *arXiv preprint arXiv:2311.18451*, 2023.

[30] Shun Lu, Yu Hu, Peihao Wang, Yan Han, Jianchao Tan, Jixiang Li, Sen Yang, and Ji Liu. Pinat: A permutation invariance augmented transformer for nas predictor. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 8957–8965, 2023.

[31] Shun Lu, Jixiang Li, Jianchao Tan, Sen Yang, and Ji Liu. Tnasp: A transformer-based nas predictor with a self-evolution framework. *Advances in Neural Information Processing Systems*, 34:15125–15137, 2021.

[32] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. *Advances in neural information processing systems*, 31, 2018.

[33] Jindi Lv, Yuhao Zhou, Yuxin Tian, Qing Ye, Wentao Feng, and Jiancheng Lv. Hypernas: Enhancing architecture representation for nas predictor via hypernetwork. *arXiv preprint arXiv:2509.18151*, 2025.

[34] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[35] Joe Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. In *International conference on machine learning*, pages 7588–7598. PMLR, 2021.

[36] Keith G Mills, Di Niu, Mohammad Salameh, Weichen Qiu, Fred X Han, Puyuan Liu, Jialin Zhang, Wei Lu, and Shangling Jui. Aio-p: Expanding neural performance predictors beyond image classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 9180–9189, 2023.

[37] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 7. Granada, 2011.

[38] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE, 2012.

[39] Yameng Peng, Andy Song, Haytham M Fayek, Vic Ciesielski, and Xiaojun Chang. Swap-nas: Sample-wise activation patterns for ultra-fast nas. *arXiv preprint arXiv:2403.04161*, 2024.

[40] Yijian Qin, Ziwei Zhang, Xin Wang, Zeyang Zhang, and Wenwu Zhu. Nas-bench-graph: Benchmarking graph neural architecture search. *Advances in neural information processing systems*, 35:54–69, 2022.

[41] Gresa Shala, Thomas Elsken, Frank Hutter, and Josif Grabocka. Transfer nas with meta-learned bayesian surrogates. In *The Eleventh International Conference on Learning Representations*, 2023.

[42] Yanan Sun, Handing Wang, Bing Xue, Yaochu Jin, Gary G Yen, and Mengjie Zhang. Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor. *IEEE Transactions on Evolutionary Computation*, 24(2):350–364, 2019.

[43] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(11), 2008.

[44] Wei Wen, Hanxiao Liu, Yiran Chen, Hai Li, Gabriel Bender, and Pieter-Jan Kindermans. Neural predictor for neural architecture search. In *European conference on computer vision*, pages 660–676. Springer, 2020.

[45] Colin White, Willie Neiswanger, and Yash Savani. Bananas: Bayesian optimization with neural architectures for neural architecture search. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 10293–10301, 2021.

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

[47] Shen Yan, Yu Zheng, Wei Ao, Xiao Zeng, and Mi Zhang. Does unsupervised architecture representation learning help neural architecture search? *Advances in neural information processing systems*, 33:12486–12498, 2020.

[48] Chris Zhang, Mengye Ren, and Raquel Urtasun. Graph hypernetworks for neural architecture search. *arXiv preprint arXiv:1810.05749*, 2018.

[49] Yi-Kai Zhang, Ting-Ji Huang, Yao-Xiang Ding, De-Chuan Zhan, and Han-Jia Ye. Model spider: Learning to rank pre-trained models efficiently. *Advances in Neural Information Processing Systems*, 36:13692–13719, 2023.

[50] Kaixiong Zhou, Xiao Huang, Qingquan Song, Rui Chen, and Xia Hu. Auto-gnn: Neural architecture search of graph neural networks. *Frontiers in big Data*, 5:1029307, 2022.

A Experimental Details

A.1 Search Spaces

NAS-Bench-201 [13] A cell-based search space containing 15,625 architectures. Each cell is a DAG with 4 nodes and 6 edges, where each edge can take one of 5 operations: zeroize, skip-connection, 1×1 convolution, 3×3 convolution, and avg-pooling. It provides full ground-truth performance for CIFAR-10, CIFAR-100, and ImageNet-16-120.

MobileNetV3 [19] A large-scale, chain-structured space based on the Once-for-All (OFA) [5] supernet. The searchable dimensions include kernel size $\{3, 5, 7\}$, expansion ratio $\{3, 4, 6\}$, and depth $\{2, 3, 4\}$ across 5 stages, resulting in approximately 10^{19} candidates.

A.2 Search Datasets

CIFAR-10 [24] A dataset consists of $60,000$ 32×32 color images in 10 broad classes, with $6,000$ images per class. The categories include common objects and animals, such as airplanes, automobiles, birds, and cats. It serves as the primary baseline for evaluating the fundamental ranking capability of NAS performance predictors.

CIFAR-100 [24] Similar to CIFAR-10 in scale and resolution, this dataset contains $60,000$ images but is divided into 100 specialized classes. Each class contains 600 images, which are further grouped into 20 superclasses, representing a higher level of label complexity and fine-grained diversity.

MNIST [9] A classic dataset of handwritten digits $(0 - 9)$ consisting of $70,000$ grayscale images of size 28×28. It represents a simple yet fundamental domain for testing the transferability of predictors to non-natural image data.

SVHN [37] A dataset contains over $600,000$ real-world digit images obtained from house numbers in Google Street View. Unlike MNIST, SVHN involves cluttered backgrounds and significant variations in appearance, posing a more challenging digit recognition task.

FGVC-Aircraft [34] A fine-grained dataset contains $10,000$ images across 100 aircraft model variants. It is characterized by highly similar visual structures and subtle discriminative features (e.g., engine placement or window configurations), requiring the predictor to capture intricate architectural cues for effective performance estimation.

Oxford-IIIT Pet [38] A fine-grained dataset featuring 37 categories of cats and dogs with approximately 200 images per class. It tests the predictor's ability to distinguish between subtle inter-class differences (e.g., different breeds).

ImageNet-1K [8] A large-scale dataset containing approximately 1.28 million training images and $50,000$ validation images across $1,000$ diverse object categories. The images feature significant variations in scale, orientation, and background, covering a vast spectrum of real-world objects and animal species.

ImageNet-16-120 [7] A downsampled version of the ImageNet dataset, containing $151,700$ images across 120 categories. All images are resized to a resolution of 16×16 pixels. It maintains the diversity of the original ImageNet while offering a more computationally efficient benchmark for evaluating ranking consistency.

A.3 Meta-training Sample Set

To enable the cross-dataset predictive capabilities of ACDP, we leverage the meta-training dataset $T = \{(d_i, a_i, y_i)\}_{i=1}^{t}$ provided by MetaD2A [25], where t denotes the total number of triplet samples. Each sample in T is a triplet consisting of a dataset $d_i \in D$, an architecture $a_i \in A$, and its corresponding ground-truth performance y_i. We utilize two distinct benchmarks from this set:

NAS-Bench-201: For this search space, each dataset $d_i \in D$ is a 20-class subset sampled from ImageNet-1K (downsampled to 32×32 pixels). The architectures are sourced from a high-quality candidate pool within NAS-Bench-201, which includes the top 5,000 models on ImageNet-16-120 and architectures discovered by GDAS [12]. In total, we utilize 4,630 triplets, where architectures are randomly paired with datasets from this candidate pool.

MobileNetV3: This large-scale benchmark introduces higher complexity, with each dataset $d_i \in D$ containing class counts ranging from 50 to 1,000 sampled from ImageNet-1K (downsampled to 32×32 pixels). The architectures are randomly sampled from the expansive MobileNetV3 search space, covering diverse kernel sizes, expansion ratios, and network depths to ensure structural variety. In total, we utilize 4,310 triplets for training.

A.4 Architectures Encoding for Different Search Spaces

We employ a Bidirectional Graph Convolutional Network (BiGCN) to extract architectural features, using tailored encoding protocols for the topological differences between the two spaces.

NAS-Bench-201 Following Neural Predictor [44], we convert each cell into a graph representation: For adjacency matrix, each cell is an 8×8 matrix (4 nodes, 6 edges, plus input/output). We implement a pruning logic where none or skip_connect edges trigger a dynamic rewiring to bypass redundant nodes. For node features, edge operations are mapped to 5D one-hot vectors and assigned to their destination nodes. We use a 6-layer Bi-GCN to propagate features, followed by global graph pooling to produce the final embedding.

MobileNetV3 For the sequential MBV3 space, we adapt the GCN framework to a chain-like topology: For the adjacency matrix, we utilize a fixed 22×22 linear chain graph, representing 20 searchable MBConv blocks plus input/output nodes. For node features, each block's configuration is a concatenated 9D vector consisting of one-hot encoded kernel size (ks), expansion ratio (e), and stage depth (d). This vector is padded to 15D to satisfy the GCN input requirements. Consistent with the 201 space, we employ a 6-layer Directed GCN. The adjacency matrix is normalized with self-loops ($\hat{N} = N + I$) to ensure stable information flow across the 22 nodes.

A.5 Elastic Dataset Traits for Different Search Spaces

To ensure the generalizability of ACDP across diverse architectural paradigms, we instantiate the elastic dataset traits with space-specific configurations for NAS-Bench-201 and MobileNetV3.

NAS-Bench-201 For intrinsic expressivity (H_{det}), following Eq. (4), we compute the activation overlap matrix M via a data-dependent forward pass with 128 images. This process is training-free and requires only a single inference step to capture the sub-network's representational diversity. In our implementation, M is computed by aggregating binary activation patterns across all ReLU layers to quantify the representational diversity. For low-fidelity accuracy (Acc_5), since each meta-task involves a 20-class subset of ImageNet-1K, we utilize the entire data of the assigned task for training to ensure the stability of the rapid performance feedback. The training for the NAS-Bench-201 space is strictly aligned with the official benchmark specifications: architectures are trained for 5 epochs following the official training protocol (Adam optimizer, lr $= 0.001$, batch $= 128$, label smoothing $= 0.1$). Finally, the discovered architectures are trained to obtain their ground-truth performance (SGD optimizer, lr $= 0.1$, momentum $= 0.9$, weight decay $= 0.0005$, cosine annealing scheduler, batch $= 256$). The training duration is 50 epochs for MNIST and 200 epochs for other datasets.

MobileNetV3 For intrinsic expressivity (H_{det}), we extend the activation analysis to MobileNetV3-specific non-linearities, including ReLU6 and Hard-Swish. The activation overlap matrix M is computed by aggregating binary activation patterns across these layers via a data-dependent forward pass with 128 images. This process is training-free with a single inference step. For low-fidelity accuracy (Acc_5), given that the datasets in MobileNetV3 can scale up to the full 1,000 ImageNet classes, we perform sparse sampling of 10 images per class for this adaptation. Meanwhile, we implement a hierarchical strategy to bridge the gap between training and practical search. During the training phase, since the architecture performance in the reference database [25] is not based on supernet inheritance, we train the candidate architectures from scratch for 5 epochs on the target task (SGD optimizer, lr $= 0.01$, momentum $= 0.9$, weight decay $= 0.0001$, cosine annealing scheduler, label smoothing $= 0.1$, batch $= 32$) This ensures that the dataset traits are aligned with the performance distribution of the training samples. During the search phase, to efficiently evaluate unseen datasets, we leverage the pre-trained OFA supernet [5] to facilitate feature extraction. To ensure compatibility with the OFA supernet, all dataset images are resized to 224×224 pixels prior to training. The architectures are trained for 5 epochs following the official OFA adaptation protocol (SGD optimizer, lr $= 0.01$, momentum $= 0.9$, weight decay $= 0.0005$, cosine annealing scheduler,

batch $= 96$). Finally, the architectures discovered during the search process are evaluated by inheriting weights from the OFA and training for 150 epochs to obtain their ground-truth performance.

Furthermore, we conduct an extensive evaluation of the fusion weights for Acc_5 and H_{det}. While a balanced 1:1 summation yields competitive results across both NAS-Bench-201 and MobileNetV3, we observe that $0.1 \cdot H_{det} + Acc_5$ further enhances prediction in MobileNetV3. This phenomenon can be attributed to the disparate information density of the two traits: in the astronomically large MobileNetV3 space (10^{19}), H_{det} serves as a coarse-grained prior of architectural capacity, whereas Acc_5 provides fine-grained, task-specific empirical grounding. By appropriately scaling H_{det}, we ensure that once the search enters the high-performance region, the predictor prioritizes the more reliable low-fidelity performance feedback, effectively mitigating the noise inherent in zero-cost proxies for complex, large-scale structural evaluations.

A.6 Implementation Details and Hyperparameters

We provide the detailed architecture configurations and training hyperparameters for ACDP below.

Model Architectures The architecture encoder is implemented as a 6-layer Bidirectional Graph Convolutional Network (Bi-GCN) to effectively capture the structural dependencies of neural candidates. The Dynamic Hypernetwork ($\mathcal{H}_{\text{Dyn}}$) and the Architecture-aware Extractor ($\mathcal{M}$) are constructed using 2-layer and 3-layer Multi-Layer Perceptrons (MLPs), respectively, providing a balanced capacity for meta-knowledge extraction and performance projection.

Training Specifications We utilize the Adam optimizer with a learning rate of 1×10^{-4} and a weight decay of 1×10^{-5} to optimize the model parameters. To stabilize the training process, we employ a Cosine Annealing Learning Rate Scheduler with a maximum number of epochs $T_{\text{max}} = 50$.

Loss Function The margin m in our pairwise margin ranking loss is fixed at 0.5. This setting ensures that the model develops heightened ranking discriminability, forcing the predicted performance gap between any two architectures to align with their true performance hierarchy with a safety margin.

B Baselines

In this section, we provide detailed descriptions of the baselines compared in Section 4. These methods are categorized into single-dataset performance predictors, cross-dataset performance predictors, and representative NAS algorithms.

B.1 Single-Dataset Performance Predictors

Most existing performance predictors follow the single-dataset training paradigm, aiming to enhance predictive accuracy on a specific dataset with low supervision by optimizing model architectures or learning strategies. To provide a comprehensive evaluation, we compare ACDP against a range of classic and recent top-performing single-dataset predictors.

For foundational baselines, NAO [32] proposes a neural architecture optimization framework that maps discrete architectures into a continuous latent space to enable gradient-based search; NP [44] employs a conceptually simple regression model trained on random architecture-accuracy pairs to estimate performance; Arch2Vec [47] introduces unsupervised representation learning to decouple architecture embedding from search, clustering architectures with similar structural properties; BRP-NAS [14] enhances prediction quality by considering binary relations between architectures and adopting an iterative data selection strategy; BANANAS [45] framework couples a neural predictor with Bayesian optimization, utilizing a novel path-based encoding to improve sample efficiency; TNASP/TNASP-SE [31] utilize a Transformer-based predictor with Laplacian positional encoding and a self-evolution framework to capture both topology and temporal information; and EZNAS [1] automates the discovery of interpretable zero-cost proxies through genetic programming to alleviate expensive training routines. For recent top-performing methods, PINAT [30] enhances architecture characterization using a permutation invariance augmentation module within a Transformer-like architecture; CAP [21] develops a context-aware predictor that infers contextual structures around nodes through self-supervised tasks, achieving high precision with sparse annotations; CARL [20] applies causality-guided representation learning to separate critical causal substructures from redundant ones, addressing distribution shifts and spurious correlations; and ParZC [10] introduces parametric

Table 4: Efficiency comparison on CIFAR-10.

Method	Train Samples	Test Samples	Search Time
Single-dataset Predictor			
PINAT [30]	469 (3%)	all	>18 GPU days
CAP [21]	469 (3%)	all	>18 GPU days
CARL [20]	469 (3%)	all	>18 GPU days
ParZC [10]	156 (1%)	all	10.9 GPU days
Cross-dataset Predictor			
MetaD2A [25]	0	all	1370s
DiffusionNAG [2]	0	all	35s
ACDP-zero (Ours)	**0**	**all**	**30s**
ACDP-all (Ours)	**5-epoch for all**†	**all**	**7.4 GPU days**

†: ACDP-all uses 5-epoch probing for full samples.

zero-cost proxies using Bayesian Networks to estimate node-specific uncertainty, optimized via a differentiable Kendall's Tau loss to improve ranking consistency.

B.2 Cross-Dataset Performance Predictors

Cross-dataset performance prediction is relatively under-explored as a standalone research task. Currently, only two representative frameworks, MetaD2A [25] and DiffusionNAG [2], involve cross-dataset prediction. However, both methods focus on Neural Architecture Generation (NAG), where the predictor serves merely as an auxiliary plug-in for candidate filtering or diffusion guidance.

Specifically, MetaD2A [25] generates architectures via a cross-modal latent space and contributes the extensive (dataset, architecture, performance) triplets that serve as the primary data source for training ACDP. DiffusionNAG [2] further employs a graph diffusion model, using a predictor to steer the generation process toward task-optimal regions. In practice, both methods adopt a conventional paradigm: they extract dataset and architectural features separately and then use a naive concatenation as input to an MLP. This simple fusion strategy fails to capture the intricate architecture-aware interactions, which ACDP is designed to address.

B.3 Representative NAS Methods

To evaluate the practical utility of ACDP in guiding architecture search, we introduce several representative NAS strategies: ResNet [18] provides a landmark convolutional baseline that utilizes residual connections to facilitate the training of deep networks; RSPS [27] integrates random search with weight-sharing by training stochastically sampled sub-graphs from a weight-shared DAG; SETN [11] employs a one-shot selective sampling strategy to evaluate competitive child candidates based on validation performance; GDAS [12] leverages a Gumbel-Softmax-based differentiable sampler to optimize architecture parameters by minimizing validation loss; PC-DARTS [46] improves efficiency by partially sampling channels to reduce memory usage and search time during the gradient-based search; DrNAS [6] introduces a Dirichlet distribution to model the architecture space, enhancing the generalization; and TNAS [41] is a subsequent advancement of MetaD2A that addresses the lack of robustness across datasets by meta-learning a Bayesian surrogate model, which utilizes deep-kernel Gaussian Processes to transfer architecture knowledge to unseen tasks.

C Extended Experimental Results

In this section, we provide experimental results. This includes extended evaluations of our core experiments on additional datasets and methods. Furthermore, we provide detailed data that were omitted from the main text due to space constraints, ensuring a thorough validation of ACDP.

C.1 Efficiency Analysis on Predicted Rank

To quantify the efficiency gains of ACDP, we analyze its computational overhead on the NAS-Bench-201 search space. As shown in the Table 4, ACDP-zero requires only a few dozen seconds of wall-

clock time to complete the entire prediction process, as it does not require any architecture training or probing, yet achieves performance far exceeding cross-dataset SOTAs. To further analyze the upper bound of ACDP-zero's precision, we conducted 5-epoch probing on the complete search space containing 15,625 architectures, requiring a total of 78,125 epochs, which translates to approximately 7.4 GPU days. Notably, the end-to-end times listed in the table for single-dataset predictors include the time required for the complete training of the sample architecture. For example, traditional single-dataset SOTA methods typically require at least 250 epochs of full-label training for a small subset of architectures (comprising 156 training and 200 validation architectures, totaling 356); this accumulates to 89,000 epochs, or approximately 10.9 GPU days. Even in this most conservative scenario involving full-space probing, ACDP-all still reduces computing resource consumption by over 3.5 GPU days while achieving superior prediction accuracy ($\tau = 0.759$ vs. 0.706).

All experiments are conducted on NVIDIA GeForce RTX 3090 GPUs (24GB GDDR6X memory).

C.2 Correlation Analysis on Predicted Rank on CIFAR-100

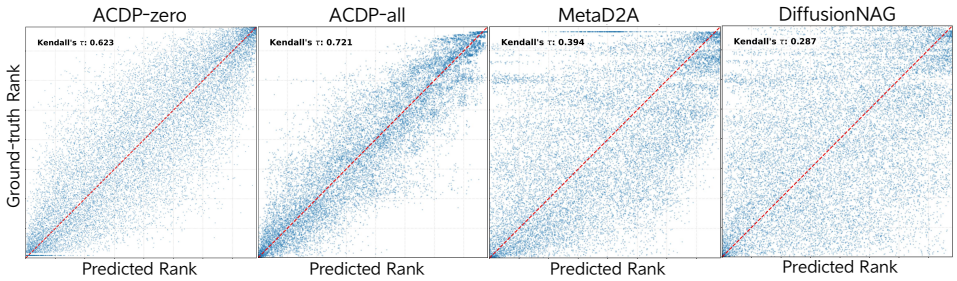

Figure 9: Rank correlation landscape on the CIFAR-100.

Complementing the rank correlation analysis presented in Section 4.2.1 of the main text, we extend the full-spectrum evaluation to the more complex CIFAR-100 domain to verify the scalability of ACDP's ranking calibration. As illustrated in Figure 9, ACDP-zero has Kendall's τ of 0.623, and ACDP-all maintains a high τ of 0.721, with its predictive distribution closely adhering to the diagonal across the entire performance spectrum. In contrast, the MetaD2A baseline ($\tau = 0.394$) suffers from severe global dispersion, indicating a substantial loss of discriminative granularity on more challenging tasks. These results reinforce that ACDP's predictive reliability is not confined to specific datasets but stems from a robust, fundamental understanding of the architecture-performance mapping across diverse task complexities.

C.3 Cross-Dataset Task-Awareness in DiffusionNAG

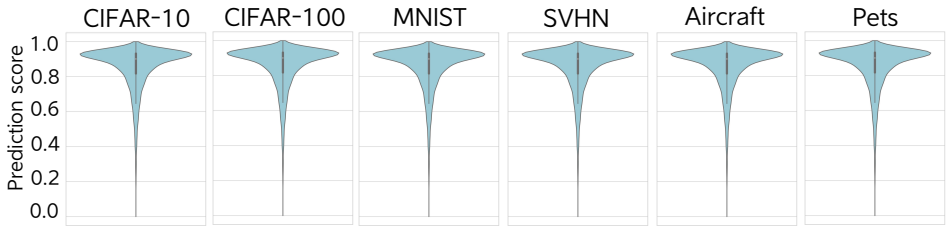

Figure 10: Predicted performance profiles: DiffusionNAG

As discussed in Section 4.2.1, we provide the predicted performance distributions of DiffusionNAG [2] across various datasets in Figure 10. Consistent with MetaD2A, DiffusionNAG exhibits a distinct task insensitivity: the predicted profiles remain static and fail to show any "Distribution Shift" regardless of whether the target domain is simple (e.g., MNIST) or complex (e.g., Aircraft). This further validates that without explicit task-aware modulation, current predictors struggle to adapt their evaluative logic to diverse dataset semantics.

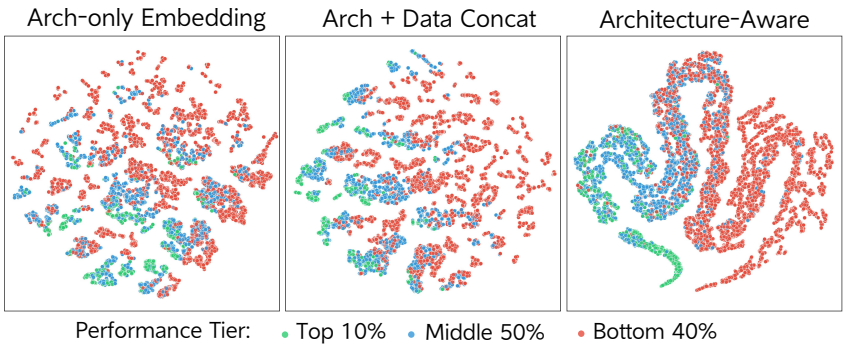

Figure 11: Latent manifold evolution on the CIFAR-100.

Table 5: Kendall's τ comparison across different methods on NAS-Bench-201.

Input Signal	Dataset	Signal Only	Concat	MLP	Arch-aware
H_{det}	CIFAR-10	0.544	0.553	0.589	**0.633**
	CIFAR-100	0.534	0.556	0.588	**0.623**
Acc_5	CIFAR-10	0.644	0.648	0.661	**0.708**
	CIFAR-100	0.421	0.429	0.432	**0.458**
$H_{det} + Acc_5$	CIFAR-10	0.715	0.708	0.722	**0.759**
	CIFAR-100	0.698	0.694	0.702	**0.721**

C.4 Qualitative Representation Visualization on CIFAR-100

Complementing the visualization analysis in Section 4.3.2 of the main text, we further investigate the latent representation space on the more complex CIFAR-100 dataset to observe the internal "disordered-to-aware" evolution under heightened task complexity. As shown in Figure 11, the transition from chaotic entanglement in the Arch-only configuration to performance-driven clustering in the full ACDP-zero configuration is consistently observed. Even on CIFAR-100, top-performing architectures are effectively mapped to distinct, island-like manifolds with minimal overlap between performance tiers. The consistency of this manifold reconstruction across diverse datasets, from CIFAR-10 to the more challenging CIFAR-100, further validates the universality of ACDP's internal representation mechanism.

C.5 Extended Ablation Study on Input Signals and Hybrid Indicators

To comprehensively evaluate the impact of different input signals and isolate the contribution of the proposed architecture-aware predictor, we conduct extended ablation studies across various proxy signals (including H_{det}, Acc_5, and their hybrid combination) under identical architecture encoders and parameter budgets.

As summarized in Table 5, we compare four configurations: 1) Signal Only; 2) Static Concat; 3) MLP; and 4) Arch-aware.

The results show that using the signal alone already provides meaningful architecture-dependent information (e.g., achieving Kendall's τ of 0.544 and 0.534 for H_{det} on CIFAR-10 and CIFAR-100, respectively). Notably, while Acc_5 alone yields a strong baseline correlation on CIFAR-10 (0.644), it experiences a marked drop on CIFAR-100 (0.421), reflecting the inherent domain sensitivity of standalone proxy signals under cross-dataset shifts. When combining H_{det} and Acc_5 into a hybrid indicator, the baseline ranking correlation rises substantially (0.715 on CIFAR-10 and 0.698 on CIFAR-100), demonstrating strong signal complementarity.

More importantly, compared with using the signal alone, the proposed architecture-aware mechanism consistently improves the ranking correlation across all tested signal configurations—boosting H_{det}, Acc_5, and hybrid signals to (0.633/0.623), (0.708/0.458), and (0.759/0.721) respectively. This confirms that the performance enhancement does not solely rely on a specific raw proxy signal.

Table 6: Ablation comparison of final search accuracy (%) across four heterogeneous datasets.

Method	MNIST	SVHN	Aircraft	Pets
Signal Only	99.66±0.12	96.27±0.06	57.01±0.21	39.98±0.08
Concat	99.68±0.07	96.33±0.04	57.04±0.17	40.12±0.06
MLP	99.67±0.08	96.35±0.01	57.03±0.11	40.35±0.07
Arch-aware (ACDP-zero)	**99.73±0.03**	**96.52±0.01**	**57.22±0.22**	**41.22±0.08**

Table 7: Global rank correlation (τ / ρ) in Nas-Bench-201 with zero-cost signals.

Method	CIFAR-10 (τ / ρ)	CIFAR-100 (τ / ρ)
SWAP [39]	0.59 / 0.79	0.61 / 0.81
ZiCo [26]	0.61 / 0.80	0.61 / 0.81
MeCo [22]	0.61 / 0.89	0.60 / 0.88
ACDP-zero	**0.63 / 0.91**	**0.62 / 0.89**

We emphasize that ACDP-zero is not designed to introduce a stronger standalone zero-cost proxy. Instead, its goal is to learn transferable architecture-performance relationships across datasets. Although some proxies or hybrid combinations may achieve higher standalone ranking correlation on individual datasets, they do not explicitly model how architecture properties transfer under dataset shifts. The proxy serves as a lightweight architecture-dependent representation, while the proposed predictor learns the cross-dataset mapping from such representations to target performance. Moreover, the proposed framework is not tied to a specific proxy and can flexibly incorporate diverse architecture-dependent signals. Therefore, the performance improvement of ACDP comes from the combination of architecture-aware cross-dataset modeling and architecture-dependent representations, rather than from simply using a stronger zero-cost proxy.

C.6 Extended Ablation Study on Heterogeneous Datasets

To further validate the effectiveness of the proposed architecture-aware mechanism on diverse, heterogeneous datasets, we supplement an ablation study across four additional datasets: MNIST, SVHN, Aircraft, and Pets. Following the same comparison settings as in Figure 7(b) and Table 2, we evaluate four configurations (Only, Concat, MLP, and Arch-aware / ACDP-zero) using the final search accuracy as the evaluation metric.

As clearly shown in Table 6, on heterogeneous datasets, the performance improvements brought by simple feature concatenation or a basic MLP are highly limited and may even encounter feature alignment bottlenecks. In contrast, our proposed architecture-aware mechanism achieves the most significant performance leaps across all heterogeneous datasets. This strongly proves that our core mechanism is not only effective on homogeneous data but also maintains exceptional generalization when processing complex cross-domain heterogeneous features.

D Additional Experimental Analysis

In this section, we provide several additional experiments not included in the original text, aimed at further evaluating the effectiveness and robustness of ACDP across diverse scenarios.

D.1 Comparison with State-of-the-Art Zero-Cost Proxies

To further evaluate the intrinsic quality of the zero-cost metrics derived by ACDP-zero, we integrate the latest zero-cost proxy methods (e.g., SWAP [39], ZiCo [26], and MeCo [22]) into our unified evaluation framework for a head-to-head comparison.

D.1.1 Global Ranking

We first conduct a global ranking experiment by calculating metric scores for all 15,625 architectures within the NAS-Bench-201 search space. Table 7 reports the Kendall's τ and Spearman's ρ rank

Table 8: Search accuracy (%) comparison in Nas-Bench-201 with zero-cost signals.

Method	CIFAR-10	CIFAR-100	MNIST	SVHN	Aircraft	Pets
SWAP [39]	94.36±0.01	73.47±0.12	99.68±0.05	96.46±0.07	56.88±0.11	38.79±0.08
ZiCo [26]	94.37±0.02	73.42±0.11	99.72±0.13	96.48±0.07	57.19±0.06	41.08±0.07
MeCo [22]	94.34±0.04	73.44±0.15	99.69±0.11	96.42±0.09	57.03±0.04	41.13±0.13
ACDP-zero	**94.37±0.00**	**73.51±0.00**	**99.73±0.03**	**96.52±0.01**	**57.22±0.22**	**41.22±0.08**

Table 9: Kendall's τ comparison when all methods access the same Acc_5 information.

Method	CIFAR-10					CIFAR-100				
Train Samples	0	$0^\dagger$	78	156	469	0	$0^\dagger$	78	156	469
CAP [21]	–	–	0.600	0.684	0.755	–	–	0.554	0.632	0.701
CAP† [21]	–	–	0.612	0.698	0.759	–	–	0.533	0.612	0.689
CARL [20]	–	–	0.656	0.706	0.806	–	–	0.578	0.645	0.718
CARL† [20]	–	–	0.642	0.691	0.813	–	–	0.562	0.623	0.701
MetaD2A [25]	0.381	0.422	–	–	–	0.394	0.399	–	–	–
DiffusionNAG [2]	0.280	0.294	–	–	–	0.287	0.296	–	–	–
ACDP (Ours)	0.633	**0.759**	–	–	–	0.623	**0.721**	–	–	–

†: Using 5-epoch probing for full samples to fine-tune.

correlation coefficients on CIFAR-10 and CIFAR-100. ACDP-zero consistently achieves the highest correlations, particularly on the more challenging CIFAR-100 dataset ($\tau = 0.62$ / $\rho = 0.89$). This suggests that ACDP-zero captures more reliable performance cues than existing human-designed or statistics-based zero-cost proxies.

D.1.2 Practical Search

We further assess the practical utility of these metrics by performing a search. Substantially, we score all candidates globally and directly select the architecture with the highest score for evaluation. As shown in Table 8, ACDP-zero identifies architectures that reach or exceed the performance of top-tier proxies across all six datasets.

These results demonstrate that the zero-cost indicators H_{det} learned by ACDP-zero possess superior discriminative power, effectively surpassing current state-of-the-art zero-cost proxies in both theoretical ranking and practical search tasks.

D.2 Fairness Evaluation with Target-Dataset Probing

To investigate whether the performance gains of ACDP-all stem purely from the additional 5-epoch probing (Acc_5) or still from its superior architecture-aware mechanism, we conduct controlled experiments by integrating Acc_5 into all baseline methods. Specifically, we selected the latest open-source single/cross-dataset predictors. After fully training them following their original methods, we further fine-tuned them using Acc_5 of the target architectures before the final evaluation. This setup strictly ensures that all baselines obtain the exact same target dataset information as ACDP-all.

D.2.1 Global Ranking

Before evaluating the Nas-Bench-201 space, we fine-tune the trained baseline predictors using Acc_5 labels of all candidates. We then use these retrained models to output final predictions. As shown in Table 9, although Acc_5 provides a reliable baseline on simple datasets like CIFAR-10 ($\tau \approx 0.65$), it introduces significant noise on complex domains like CIFAR-100 ($\tau \approx 0.45$). Consequently, fine-tuning single-dataset predictors yields marginal gains on CIFAR-10 but causes a performance drop on CIFAR-100. This exposes the inherent inability of traditional predictors to align and filter low-fidelity signals. In contrast, ACDP-all achieves a Kendall's τ of 0.759 and 0.721, respectively, significantly surpassing all fine-tuned baselines. This proves that ACDP's advantage arises from its architecture-aware module, which effectively maximizes the utility of low-fidelity signals.

Table 10: Search accuracy (%) in Nas-Bench-201 with Acc_5 probing.

Method	CIFAR-10	CIFAR-100	MNIST	SVHN	Aircraft	Pets
CAP† [21]	94.36±0.01	73.37±0.04	–	–	–	–
CARL† [20]	94.37±0.03	73.45±0.08	–	–	–	–
MetaD2A† [25]	**94.37±0.00**	**73.51±0.00**	99.67±0.11	96.48±0.06	56.99±0.12	41.21±0.09
DiffusionNAG† [2]	**94.37±0.00**	**73.51±0.00**	99.71±0.04	96.48±0.07	57.03±0.08	39.22±0.05
ACDP-10% (Ours)	**94.37±0.00**	**73.51±0.00**	**99.74±0.02**	**96.57±0.01**	**57.35±0.28**	**41.26±0.18**

†: Using 5-epoch probing for Top-10% architectures to fine-tune.

Table 11: Search accuracy (%) in MobileNetV3 with Acc_5 probing.

Method	CIFAR-10	CIFAR-100	MNIST	SVHN	Aircraft	Pets
MetaD2A† [25]	97.77±0.04	85.62±0.12	99.58±0.06	96.81±0.08	77.78±0.03	94.07±0.11
Diff-NAG† [2]	97.92±0.06	85.98±0.05	99.58±0.13	96.90±0.21	77.64±0.11	94.27±0.07
ACDP-10% (Ours)	**98.04±0.04**	**87.45±0.12**	**99.66±0.08**	**97.11±0.12**	**79.57±0.02**	**94.77±0.11**

†: Using 5-epoch probing for Top-10% architectures to fine-tune.

To further validate practical utility, all baselines follow our hierarchical strategy: they first use initial rankings to filter the top 10% candidates, then utilize Acc_5 for fine-tuning to select the final optimal architecture. Since single-dataset predictors require complete target dataset samples, they are only evaluated on CIFAR-10/100 in Nas-Bench-201, where full labels exist.

As shown in Table 10, ACDP-all consistently matches or exceeds the SOTA performance across all six datasets. A critical observation is the operational dependency of single-dataset predictors (e.g., CAP [21], CARL [20]). These methods inherently rely on a significant volume of ground-truth labels from the target domain to train their weights from scratch. Consequently, their evaluation is strictly limited to the CIFAR tasks within NAS-Bench-201, where full-spectrum labels are available. In stark contrast, ACDP bypasses this constraint by leveraging its pre-trained cross-dataset knowledge, enabling it to generalize to non-CIFAR domains (MNIST, SVHN, Aircraft, and Pets) where target-domain training data is absent.

In the more challenging MobileNetV3 space (Table 11), ACDP-all still demonstrates a clear lead. These results underscore that ACDP-all's superiority is not a byproduct of information asymmetry. Even when all methods have access to the same probing data, ACDP-all's ability to dynamically transform raw dataset traits into task-specific representations allows it to maintain a significant edge in complex search environments.

D.3 Extensibility to Non-Vision Domains: NAS-Bench-Graph

To evaluate the generalizability of ACDP beyond convolutional neural networks, we extend our evaluation to the NAS-Bench-Graph [40] benchmark. This benchmark encompasses 9 diverse graph datasets spanning citation networks and e-commerce platforms. Leveraging the epoch-wise data provided by the benchmark, we utilized low-fidelity performance signals (Acc_5) in it. We first retrained the ACDP-zero/ACDP-all predictor using 5,000 architectural samples from 5 source graph datasets. We then assessed its search performance on the remaining 4 unseen target datasets (PubMed, Physics, Computer, and arXiv). We compared ACDP-zero and ACDP-10% against a consolidated SOTA baseline. This baseline represents the upper-bound performance of established Graph-NAS methods, all of which were allowed to fully train 2% of the search space (524 full-training samples).

As illustrated in Table 12, even ACDP-zero, which requires no target-dataset probing, consistently outperforms the SOTA baselines that rely on full-training evaluations of 524 architectures. When further refined with hierarchical searching, the performance climbs higher on ACDP-10%.

These findings highlight that ACDP's performance advantage is not domain-specific. By dynamically transforming dataset-specific traits into architecture-aware features, ACDP achieves a "Train Once, Generalize Everywhere" capability, offering a highly efficient and accurate search paradigm for diverse architectural domains.

Table 12: Search accuracy (%) comparison on NAS-Bench-Graph target datasets.

Method	PubMed	Physics	Computer	arXiv
GNAS* [16]	77.79±0.02	92.43±0.04	84.74±0.20	72.00±0.02
Auto-GNN* [50]	77.69±0.16	92.42±0.16	84.53±0.14	72.13±0.03
EA* [3]	77.96±0.12	92.22±0.08	84.29±0.29	71.91±0.06
RL* [23]	77.96±0.09	92.48±0.03	84.90±0.19	72.13±0.05
ACDP-zero	78.16±0.02	92.88±0.11	85.25±0.12	72.22±0.05
ACDP-10%	**78.44±0.05**	**93.12±0.13**	**86.04±0.06**	**72.48±0.07**

*: All baselines with 2% space training (524 full-training samples).

Table 13: Performance comparison across six unseen datasets on NAS-Bench-201 search space.

Dataset	MetaD2A	MetaD2A + ACDP-zero	DiffusionNAG	DiffusionNAG + ACDP-zero
CIFAR-10	**94.37±0.00**	**94.37±0.00**	**94.37±0.00**	**94.37±0.00**
CIFAR-100	73.44±0.06	**73.51±0.00**	**73.51±0.00**	**73.51±0.00**
MNIST	99.69±0.14	**99.73±0.02**	99.62±0.08	**99.72±0.12**
SVHN	96.34±0.21	**96.55±0.14**	96.12±0.19	**96.43±0.22**
Aircraft	57.89±0.55	**58.04±0.06**	58.07±0.08	**58.25±0.15**
Pets	41.22±0.18	**42.05±0.08**	41.65±0.15	**42.12±0.24**

D.4 Prediction Engine for Generative NAS

In this section, we conduct a novel supplementary experiment to further validate the effectiveness of ACDP in enhancing NAG tasks. As discussed in Section 2.1 of the main text, existing state-of-the-art cross-dataset NAS benchmarks, such as MetaD2A and DiffusionNAG, primarily focus on the construction of architecture generators, while their internal performance predictors serve merely as auxiliary "plugins" to guide the generation process. To explore whether ACDP can function as a more powerful and universal "prediction engine" to bolster these generative frameworks, we designed a plug-and-play replacement experiment where the original prediction modules of MetaD2A and DiffusionNAG are substituted with ACDP-zero. This allows us to evaluate the impact of architecture-aware mechanisms on the overall search efficiency of NAG systems.

Due to the fundamental differences in prediction mechanisms, we adopt distinct integration strategies for the two frameworks. In DiffusionNAG, the dataset feature input of its original predictor is static (i.e., for a given dataset, the input feature embedding remains fixed regardless of the architecture). This allows the generator to utilize the gradient ∇_G of the predictor to provide continuous guidance for the architecture representation G. However, ACDP-zero utilizes "architecture-aware" dataset features, meaning that even for the same dataset, the input features dynamically evolve depending on the architecture being evaluated. This inherent dependency makes it impossible to directly calculate the gradient with respect to G to guide the diffusion process. Consequently, for DiffusionNAG, we adopt the "Unconditional Sampling + Re-ranking" (Uncond. + Sorting) strategy as described in its original work: we first generate a pool of candidate architectures unconditionally and then employ ACDP-zero for high-precision re-ranking. In contrast, MetaD2A naturally follows a "generate-then-predict" pipeline, allowing ACDP-zero to be seamlessly integrated as a plug-and-play replacement for its original evaluator. We conducted these replacement experiments on both the NAS-Bench-201 and MobileNetV3 search spaces.

For both NAS-Bench-201 and MobileNetV3 search spaces, we follow a standardized evaluation procedure: each generator first produces a candidate pool of 500 architectures, from which the corresponding predictor selects the Top-1 for full training and evaluation. To ensure a strictly fair comparison, all methods were retrained from scratch using the same dataset. Notably, since the full triplet dataset used by MetaD2A on MobileNetV3 is not publicly available, we retrained the predictors of MetaD2A and DiffusionNAG using the other 4,130-triplet subset provided by the MetaD2A official repository, consistent with the settings in Section 4.2.2. This ensures that all competing methods are evaluated under identical training data scales, thereby providing a transparent reflection of each model's inherent capability in architecture performance discrimination.

NAS-Bench-201 As shown in Table 13, within the NAS-Bench-201 search space, by replacing the original performance estimation module with ACDP-zero, the search results for MetaD2A and Diff-

Table 14: Performance comparison across six unseen datasets on the MobileNetV3 search space.

Dataset	MetaD2A	MetaD2A + ACDP-zero	DiffusionNAG	DiffusionNAG + ACDP-zero
CIFAR-10	97.41±0.12	**97.56±0.18**	97.54±0.12	**97.62±0.14**
CIFAR-100	86.14±0.35	**87.17±0.12**	86.71±0.28	**87.25±0.19**
MNIST	99.66±0.06	**99.72±0.12**	99.65±0.14	**99.70±0.12**
SVHN	96.75±0.18	**97.12±0.14**	96.82±0.15	**97.15±0.14**
Aircraft	77.52±0.62	**78.72±0.44**	77.53±0.55	**78.22±0.18**
Pets	93.95±0.21	**94.22±0.13**	94.12±0.38	**94.75±0.09**

usionNAG achieved significant accuracy improvements across all six unseen datasets. This demonstrates that ACDP is not only an accurate predictor but also a key plugin for improving the efficiency of NAG architecture generation.

MobileNetV3 In the more challenging MobileNetV3 experiments with a larger search space, the results (see Table 14) further validate the general enhancement effect of ACDP. This strongly demonstrates that the performance leap brought about by ACDP is not due to random sampling error but rather because ACDP provides performance ranking signals that are more discriminative than the original baseline.

E Limitation

While ACDP demonstrates superior cross-dataset generalization and efficiency, we acknowledge several limitations and provide a detailed discussion regarding their implications.

Computational Constraints While 5-epoch probing (Acc_5) improves precision, it can be resource-intensive. To address this, ACDP offers dual-mode flexibility: in extreme compute-limited scenarios, it can revert to ACDP-zero, which requires no target-domain training while maintaining state-of-the-art cross-dataset performance.

Extensibility Our primary evaluation focuses on vision tasks. However, supplementary experiments on NAS-Bench-Graph in Appendix D.3 (covering 9 graph datasets and operators like GCN/GAT) confirm that ACDP effectively generalizes to non-vision domains. Results show it consistently achieves SOTA accuracy with significantly lower search costs.

Pre-training Cost Collecting diverse source data for pre-training is a one-time investment. This upfront cost is justified by the long-term efficiency of a "Train Once, Generalize Everywhere" engine, which eliminates the need for repeated, expensive training on every new unseen dataset.

F Societal Impact

This work promotes green and flexible AI. By introducing a cross-dataset paradigm with a flexible hierarchical strategy, ACDP allows users to autonomously balance precision and cost. This reduces NAS computation from days to seconds, significantly lowering the carbon footprint. Furthermore, by minimizing hardware barriers, our framework enables researchers with limited resources to conduct high-performance architecture searches, fostering global inclusivity and innovation.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main contributions are clearly summarized in the Introduction, and our claims are supported by the experimental results.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have included a dedicated "Limitation" section in the Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: This paper is primarily empirical and focuses on the algorithmic framework and experimental validation of the proposed ACDP predictor. It does not contain formal theoretical theorems or proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: This paper fully discloses all essential information required to reproduce the main experimental results, including detailed descriptions of the methodology, experimental protocols, and parameter settings provided in the main text and the Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide the complete source code and necessary data configurations via a public GitHub repository (`https://github.com/Thorn-222/ACDP`). Detailed instructions on environment setup are included in the repository.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide comprehensive implementation details in the Appendix A, including dataset descriptions, hyperparameter configurations, and experimental setups, to ensure the findings can be reproduced.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The main experimental results are reported with standard deviations calculated over 3 independent runs with different random seeds, ensuring the statistical significance of our works.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the hardware specifications (e.g., GPU models) and the end-to-end latency for each experimental run in the Appendix C.1 to ensure transparency regarding the computational cost and time of execution.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in this paper fully adheres to the NeurIPS Code of Ethics and does not involve any sensitive data, human subjects, or harmful applications.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We discuss the broader impacts of our work in the Appendix F.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The proposed ACDP framework is a foundational research work on neural architecture search and does not involve high-risk models such as large-scale generative models or sensitive scraped datasets. Therefore, it does not pose a significant risk for misuse.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We have properly cited all datasets and benchmarks (e.g., NAS-Bench-201, CIFAR) in the main text and references. We strictly respect their original licenses. A formal Acknowledgments and Disclosure of Funding section has been included in this camera-ready version.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We introduce a new implementation of the ACDP predictor. The source code, along with a detailed README containing environment setup and usage instructions, is provided in the repository (`https://github.com/Thorn-222/ACDP`).

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve any crowdsourcing experiments or research conducted with human subjects. All experiments were performed using automated computational frameworks and publicly available datasets.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This research does not involve human subjects, personal data, or any experiments requiring institutional ethical approval. All evaluations were conducted on standard, publicly available machine learning benchmarks.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core methodology and scientific contributions of this research do not involve LLMs as original or non-standard components. LLMs were only utilized for minor text editing and formatting purposes, which does not impact the core research methodology.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.