ZAPS: Zero-Cost Active Proxy Search for Neural Architecture Search

Hassan Touayouch[1,2] Rabie Najem[2] Mohammed Benjelloun[2]

[1]ENSICAEN, École Nationale Supérieure d'Ingénieurs de Caen, France

[2]Université de Mons (UMONS), Mons, Belgium

Abstract

Neural Architecture Search (NAS) automates the design of neural networks but faces the prohibitive cost of evaluating each candidate architecture through full training. Zero-cost proxies estimate architecture quality in seconds without training, yet individual proxies remain noisy, and no existing method jointly exploits proxy signals and the topological structure of architectures within an active-learning framework. We propose ZAPS (*Zero-cost Active Proxy Search*), a four-component pipeline that addresses these limitations. ZAPS first selects a compact, non-redundant subset of proxies via PROXYFIT, a greedy anti-redundancy algorithm; initializes the search with a hybrid K-means strategy balancing exploitation and exploration; dynamically refines proxy selection using bootstrapped MRMR; and predicts architecture quality with an XGBoost ensemble fed by proxy ranks and one-hot topological encodings, guided by an Upper Confidence Bound (UCB) acquisition function. On NAS-Bench-201 with a budget of $B = 200$ evaluations, ZAPS identifies **53.8 %** of the top-100 architectures on CIFAR-10 and **66.8 %** on CIFAR-100 (P@100), while maintaining a Regret of **0.04 %** on CIFAR-10. ZAPS outperforms comparable-budget methods — REA, BANANAS, and TPE — on both metrics evaluated on CIFAR-10 and CIFAR-100 of NAS-Bench-201.

1 Introduction

Designing high-performing neural networks typically requires deep expertise and intensive manual experimentation. Neural Architecture Search (NAS) aims to automate this process by exploring a predefined space of candidate architectures and selecting the best one for a given task [Elsken et al., 2019]. However, evaluating each candidate requires full training, amounting to thousands of GPU hours to cover a typical search space. This computational bottleneck has spurred the development of proxy-based evaluation methods.

Zero-cost proxies. Zero-cost proxies are lightweight metrics computed at initialization — with no training at all — using only the network weights and a single forward or backward pass [Mellor et al., 2021]. They reduce evaluation time from several hours to a few seconds. Each proxy captures a different characteristic of the network (expressivity, gradient flow, complexity, etc.). The natural solution is therefore to combine them to cover a wider spectrum of characteristics. However, some proxies are strongly correlated — for example, `flops` and `params` exhibit $\rho = 0.996$ on NAS-Bench-201 / CIFAR-10 — making their naive combination redundant.

Gap in the literature. Prior work relies either on a single proxy [Mellor et al., 2021, Tanaka et al., 2020, Li et al., 2023], or naively combines several proxies via rank products [Abdelfattah et al., 2021], without accounting for their redundancy or relevance to the target accuracy. Furthermore, proxy-based methods ignore the structural encoding of the architecture — the one-hot representation of its operations and connections — which carries complementary topological information that proxies cannot capture. Surrogate-based NAS methods that use architecture encodings [White et al., 2021] operate without zero-cost proxies, requiring larger evaluation budgets.

Our approach. We propose ZAPS, a pipeline that bridges this gap by combining zero-cost proxies with the topological encoding of architectures within an active learning loop. The central intuition is that proxy scores and architecture structure are complementary: proxies reflect functional quality at initialization, while the encoding captures the graph structure that determines expressive capacity. Feeding both into a surrogate model produces substantially better predictions than either alone.

Contributions. Our main contributions are:

1. **PROXYFIT**: an offline greedy proxy selection algorithm that maximizes correlation with true accuracy while enforcing an anti-redundancy constraint (Section 3.1).

2. **Hybrid K-means initialization**: a structured strategy for collecting an initial set of architectures to evaluate, balancing exploitation of high-proxy regions and exploration of the full space (Section 3.2).

3. **Bootstrapped MRMR**: a dynamic proxy re-selection mechanism applied at each active learning iteration, adapting to the growing labeled set (Section 3.3).
4. **XGBoost ensemble with UCB acquisition**: a bootstrapped surrogate ensemble trained on features combining proxy ranks and topology, guided by an Upper Confidence Bound acquisition (Section 3.4).
5. A rigorous evaluation on NAS-Bench-201 (200 seeds, three datasets) demonstrating that ZAPS outperforms comparable-budget methods REA [Real et al., 2019], BANANAS [White et al., 2021], and TPE on both primary metrics at moderate budget ($B = 200$).

Section 2 reviews related work. Section 3 describes the ZAPS pipeline. Section 4 evaluates ZAPS experimentally. Section 5 concludes.

2 Related Work

Classical NAS. Early NAS methods use reinforcement learning [Zoph and Le, 2017] or evolutionary algorithms [Real et al., 2019], requiring thousands of GPU days. DARTS [Liu et al., 2019] introduces a differentiable relaxation of the architecture space, reducing cost to a few GPU days, but suffers from instability and cannot be applied to tabular benchmarks. All these methods require full training per evaluation.

Zero-cost proxies. NASWOT [Mellor et al., 2021] measures the rank of the activation kernel matrix to assess architecture diversity. Synflow [Tanaka et al., 2020] computes a gradient flow metric that avoids layer collapse. SNIP [Lee et al., 2019] measures connection saliency at initialization. ZiCo [Li et al., 2023] captures gradient stability across layers. Despite their efficiency, each of these proxies captures only one aspect of architectural quality and behaves inconsistently across datasets and search spaces. NAS-Bench-Suite-Zero [Zela et al., 2022] provides a unified benchmark of precomputed proxy scores across multiple search spaces — we rely directly on this resource.

Proxy ensembles and composite proxies. The rank product [Abdelfattah et al., 2021] aggregates individual proxy rankings without accounting for redundancy or relevance. EZNAS [Akhauri et al., 2022] learns composite operators over proxy statistics via evolutionary search. LPZero [Dong et al., 2024] extends this idea to architecture search for language models. These methods optimize a single composite score but do not integrate proxies into a surrogate model with structural features. SuperProxy [Zhu et al., 2022] learns a weighted combination of proxies but requires a labeled set, eliminating the zero-shot advantage. In contrast, PROXYFIT selects proxies based on individual quality and mutual redundancy, then feeds them into a surrogate that also integrates topological structure.

Surrogate-based NAS. BANANAS [White et al., 2021] encodes architectures as path encodings and trains an ensemble surrogate for Bayesian optimization. Unlike ZAPS, it does not leverage zero-cost proxies and requires a larger initial labeled set. NASBOWL [Ru et al., 2021] uses a graph-kernel surrogate but is computationally expensive. ZAPS is the first method to combine proxy ranks and one-hot topology in a joint surrogate under a strict budget.

Active learning for NAS. Active learning methods [Shen et al., 2021] select the most informative architectures to evaluate first, relying solely on architecture encodings without zero-cost proxies. ZAPS integrates active learning via UCB acquisition on a proxy-augmented surrogate.

3 Method

Problem formulation. Let $\mathcal{A}$ be an architecture search space of size N. Each architecture $a \in \mathcal{A}$ is associated with a true accuracy $f(a)$, obtainable only through full training (expensive). Given a budget B (number of allowed full evaluations), the primary objective is to identify the set $\mathcal{T}_{100}$ of the top-100 architectures, measured by $P@100 = |\hat{\mathcal{T}}_{100} \cap \mathcal{T}_{100}|/100$, where $\hat{\mathcal{T}}_{100}$ is the top-100 predicted by the surrogate. We also measure **Regret** $= (f^* - f_{\text{best}})/f^* \times 100$, where $f^* = \max_{a \in \mathcal{A}} f(a)$ is the optimal accuracy and f_{best} is the best accuracy observed among the B evaluated architectures.

Overview. ZAPS operates in four sequential steps, illustrated in Figure 1: (1) offline proxy selection via PROXYFIT, (2) hybrid K-means initialization, (3) dynamic proxy re-selection via bootstrapped MRMR, and (4) prediction by a UCB-guided XGBoost ensemble.

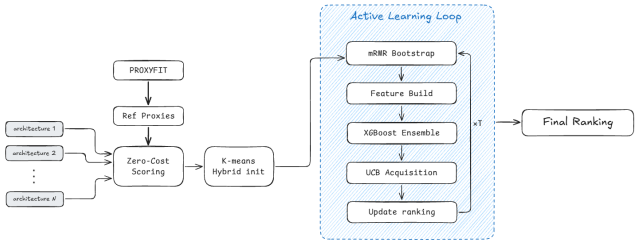

Figure 1: Overview of the ZAPS pipeline.

3.1 PROXYFIT: Anti-Redundant Proxy Selection

Let P be the set of 13 available zero-cost proxies (cf. Table 1). Naively using all of them in the surrogate introduces redundancy and noise. PROXYFIT selects a compact subset

$\Pi \subseteq P$ of size $|\Pi| = 6$ by greedily optimizing for relevance and non-redundancy.

Algorithm 1 PROXYFIT — Greedy Anti-Redundant Selection

Require: Proxy set P, redundancy threshold $\tau = 0.85$, target size $K = 6$
Ensure: Selected subset $\Pi \subseteq P$
Sort P by $|\rho(p, \text{acc})|$ in decreasing order
$\Pi \leftarrow \emptyset$
for each p in sorted P **do**
  if $\max_{q \in \Pi} |\rho(p, q)| < \tau$ **then**
    $\Pi \leftarrow \Pi \cup \{p\}$
  end if
  if $|\Pi| = K$ **then stop**
  end if
end for
return Π

The threshold $\tau = 0.85$ is set empirically. On CIFAR-10, PROXYFIT selects {`nwot`, `jacov`, `synflow`, `params`, `snip`, `grasp`}, eliminating strongly correlated proxies such as `flops` ($\rho(\texttt{flops}, \texttt{params}) = 0.996$). Notably, the selection made on CIFAR-10 transfers to CIFAR-100 and ImageNet without recalibration, suggesting that proxy complementarity is largely independent of the target dataset (see Section 4.6).

3.2 Hybrid K-means Initialization

A good initial labeled set $\mathcal{L}_0$ is crucial for surrogate quality. Pure random selection may miss promising regions, while purely greedy selection based on proxy ranks may introduce bias. We propose a hybrid strategy:

- **Biased phase** (size $n_0/2$): K-means applied to the top 30 % of architectures ranked by proxy scores. One representative per cluster is selected. Favors exploitation of high-quality regions.
- **Uniform phase** (size $n_0/2$): K-means applied to the remaining 70 %. Promotes diversity and exploration of under-explored regions.

Clustering is performed in proxy rank space. The budget allocated to initialization is $n_0 = 50$, leaving $B - n_0 = 150$ evaluations for the active learning loop at $B = 200$.

3.3 Bootstrapped MRMR

As the labeled set $\mathcal{L}_t$ grows, the relevance and redundancy of proxies may change. Rather than using a fixed subset, we re-select proxies at each iteration t according to the Maximum Relevance Minimum Redundancy (MRMR) criterion:

$$J(p) = |\rho(p, \text{acc})| - \frac{1}{|\Pi_t|} \sum_{q \in \Pi_t} |\rho(p, q)|, \tag{1}$$

where correlations are estimated on $\mathcal{L}_t$. To stabilize selection on small sets, we apply three bootstrap resamplings of $\mathcal{L}_t$ and take a majority vote. The size of Π_t grows progressively from 4 proxies to 10 proxies when $|\mathcal{L}| = 200$.

Algorithm 2 Bootstrapped MRMR

Require: Labeled set $\mathcal{L}_t$, proxies P, $B_{\text{boot}} = 3$, $k_{\min} = 4$, $k_{\max} = 10$
Ensure: Subset $\Pi_t \subseteq P$
$k \leftarrow \max(k_{\min},\ \min(k_{\max},\ \lfloor |\mathcal{L}_t|/15 \rfloor + 4))$
votes$[p] \leftarrow 0$ for all $p \in P$
for $b = 1$ **to** B_{boot} **do**
  $\mathcal{L}^b \leftarrow$ bootstrap resample of $\mathcal{L}_t$
  Greedily compute $J(p)$ from Eq. 1 on $\mathcal{L}^b$
  votes$[p]$ += 1 for each selected p
end for
return $\Pi_t \leftarrow \arg\text{top-}k_{p \in P}$ votes$[p]$

3.4 XGBoost Ensemble + UCB Acquisition

Feature representation. Each architecture a is represented by a feature vector $\phi(a)$ concatenating:

- **Proxy ranks**: for each proxy $p \in \Pi_t$, the normalized rank of a among all architectures, $\hat{r}_p(a) \in [0, 1]$.
- **Topological encoding**: a 30-dimensional binary one-hot vector encoding the operations and connections of the computation graph of a (for NAS-Bench-201).

This joint representation is the key novelty of ZAPS: no prior zero-cost method combines proxy signals and architectural structure for surrogate regression.

Bootstrap ensemble. Three XGBoost [Chen and Guestrin, 2016] regressors are independently trained on bootstrap resamplings of $\mathcal{L}_t$. Their predictions are aggregated to obtain a mean $\mu(a)$ and standard deviation $\sigma(a)$ for each unevaluated architecture $a \notin \mathcal{L}_t$.

UCB acquisition. The next batch of architectures to evaluate is selected by maximizing the Upper Confidence Bound:

$$\text{score}(a) = \mu(a) + \beta \cdot \sigma(a), \quad \beta = 0.5, \tag{2}$$

where $\mu(a)$ drives exploitation of promising architectures and $\sigma(a)$ drives exploration of architectures with uncertain estimates. The coefficient $\beta = 0.5$ is chosen to balance exploitation and exploration. At each iteration t, the b architectures with the highest UCB score are evaluated, where the batch size is:

$$b = \left\lfloor \frac{B - n_0}{T} \right\rfloor, \tag{3}$$

with $T = 2$ active learning iterations. For $B = 200$ and $n_0 = 50$, this gives $b = 75$ architectures evaluated per iteration. Algorithm 3 summarizes the full active learning loop of ZAPS.

Algorithm 3 ZAPS — Active Learning Loop

Require: Space $\mathcal{A}$, budget B, init size n_0, batch size b
Run PROXYFIT offline $\rightarrow$ subset Π_0
$\mathcal{L}_0 \leftarrow$ Hybrid K-means initialization (n_0 architectures)
Evaluate $f(a)$ for all $a \in \mathcal{L}_0$
$t \leftarrow 0$
while $|\mathcal{L}_t| < B$ **do**
    $\Pi_t \leftarrow$ Bootstrapped MRMR on $\mathcal{L}_t$
    $\phi(a) \leftarrow$ [proxy ranks under Π_t; one-hot topology]
    Train XGBoost ensemble on $\{(\phi(a), f(a))\}_{a \in \mathcal{L}_t}$
    $\mu(a), \sigma(a) \leftarrow$ ensemble mean and std
    $\mathcal{B}_t \leftarrow$ top-b unevaluated architectures by $\mu(a) + 0.5\,\sigma(a)$
    Evaluate $f(a)$ for all $a \in \mathcal{B}_t$; add to $\mathcal{L}_{t+1}$
    $t \leftarrow t + 1$
end while
return top-100 of $\mathcal{A}$ ranked by $\mu(a)$

4 Experiments

4.1 Experimental Protocol

Benchmarks. We evaluate on **NAS-Bench-201** [Dong and Yang, 2020], a search space of 15,625 architectures defined by a cell with 6 edges, each assigned one of 5 operations, on **CIFAR-10 and CIFAR-100**.

Evaluation protocol. All accuracy evaluations are table lookups in NAS-Bench-201 — no network training is performed. Proxy scores come from NAS-Bench-Suite-Zero (precomputed). ZAPS runs entirely on CPU; one seed ($B = 200$) takes approximately 5 seconds. The full set of $200 \times 3 = 600$ experiments was run in parallel.

Proxies. We use the 13 zero-cost proxies provided by NAS-Bench-Suite-Zero [Zela et al., 2022], grouped in Table 1 by family. ZAPS treats these scores as a fixed feature matrix: no proxy is recomputed at search time, and only full accuracy evaluations count toward the budget B.

Table 1: The 13 proxies from NAS-Bench-Suite-Zero [Zela et al., 2022] and their Spearman ρ on NB-201/C10.

Family	Proxies	ρ
Deterministic	`params`, `flops`, `l2_norm`, `synflow`	0.41–0.63
Activation	`nwot`, `jacov`, `epe_nas`, `zen`, `plain`	0.17–0.58
Gradient	`fisher`, `snip`, `grad_norm`, `grasp`	−0.12–0.44

Evaluation metrics. Our two primary metrics are **P@100** and **Regret**. P@100 measures the proportion of the true top-100 architectures correctly identified in the predicted top-100: $P@100 = |\hat{\mathcal{T}}_{100} \cap \mathcal{T}_{100}|/100$. Regret measures the accuracy gap between the optimal architecture and the best architecture found. We also report **Spearman correlation** ρ and **Kendall's** τ between predicted and true rankings over all 15,625 architectures. All results are averaged over **200 independent random seeds**.

Baselines. We compare ZAPS to five methods operating under the same budget B of full evaluations:

- **Random Search (RS)**: uniform selection of B architectures, with no proxy information or learning.
- **REA** [Real et al., 2019]: regularized evolution by age. A population of size 50 is maintained; at each step, a parent is selected by tournament of size 10, randomly mutated, and the oldest architecture in the population is eliminated.
- **Local Search (LS)**: greedy local search with random restarts. At each step, all direct neighbors of the current architecture (exactly one different operation across the 6 positions) are evaluated; the algorithm moves to the best neighbor if it improves accuracy, otherwise restarts from a random architecture.
- **BANANAS** [White et al., 2021]: Bayesian optimization with an ensemble of 5 MLP networks (2 layers, 20 neurons, ReLU, Adam $lr = 0.01$, 1,000 epochs). Architectures are encoded in one-hot form (30 dimensions for NAS-Bench-201); acquisition uses Thompson Sampling with full hallucination (retraining after each slot of batch size 5). The initial set comprises 10 random architectures.
- **TPE** [Bergstra et al., 2011]: Tree-structured Parzen Estimator optimization. TPE separately models the distributions of good and bad configurations, guiding search by likelihood ratio maximization. We use the Optuna implementation with default hyperparameters.

All methods are compared over the full range $B \in [100, 500]$, ensuring a fair comparison.

4.2 ZAPS Results

Figures 2 and 3 show the performance of ZAPS on CIFAR-10 and CIFAR-100 of NAS-Bench-201.

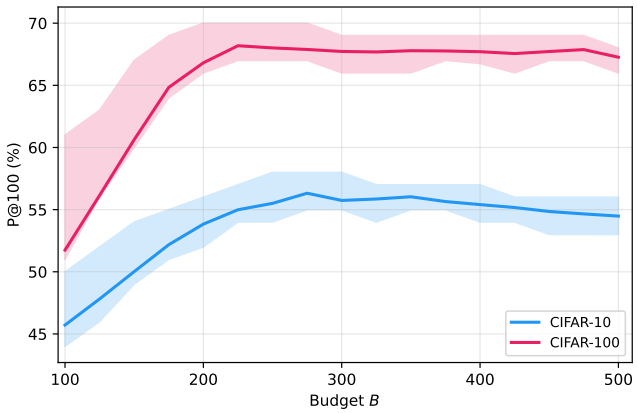

Figure 2: P@100 (%) of ZAPS as a function of budget B.

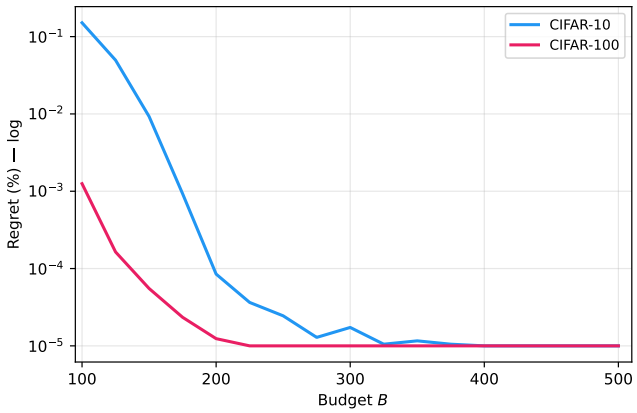

Figure 3: Regret (%) of ZAPS as a function of budget B.

ZAPS climbs rapidly: from $B = 150$, it reaches 50.0 % P@100 on CIFAR-10 and 60.6 % on CIFAR-100, and Regret drops below 0.01 % at $B = 300$ on CIFAR-100.

4.3 Comparison with Baselines

Table 2 compares ZAPS to five baselines on CIFAR-10 and CIFAR-100. At $B = 200$, ZAPS achieves 53.8 % P@100 on CIFAR-10, which is 3× more than REA (19.1 %) and 1.5× more than TPE (36.7 %). Full curves over $B \in [100, 500]$ are shown in Appendix B.

Random search, REA, and Local Search. Random search forms the floor: ≈1 % P@100 at any budget. REA and LS progress similarly at low budget, but REA gains an edge at $B = 300$ thanks to its population memory. ZAPS outperforms both at every budget.

BANANAS. The MLP surrogate of BANANAS requires a larger labeled set to converge, limiting its performance at low budget. ZAPS surpasses it on all configurations through the joint proxy + topology representation.

TPE. TPE becomes competitive at large budget: at $B = 300$, it surpasses ZAPS in P@100 on CIFAR-100 (70.5 % vs 67.7 %). At moderate budget ($B = 200$), ZAPS retains a clear advantage on CIFAR-10 (53.8 % vs 36.7 %) and CIFAR-100.

Table 2: Comparison on NAS-Bench-201 at $B = 200$ and $B = 300$. Mean ± std over 200 independent seeds. Full curves in Appendix B.

		CIFAR-10		CIFAR-100	
B	Method	P@100	Reg. (%)	P@100	Reg. (%)
200	RS	1.1±1.0	0.72±0.27	1.2±1.1	1.89±0.90
	REA	19.1±6.4	0.18±0.17	24.3±8.4	0.13±0.33
	LS	17.3±9.1	0.25±0.23	23.0±13.1	0.30±0.51
	BANANAS	2.0±1.4	0.59±0.23	2.4±1.6	1.09±0.78
	TPE	36.7±14.8	0.10±0.17	51.2±13.5	0.03±0.06
	ZAPS	**53.8±3.5**	**0.04±0.09**	**66.8±6.9**	**0.02±0.14**
300	RS	1.9±1.4	0.63±0.23	1.9±1.4	1.43±0.84
	REA	29.7±7.3	0.10±0.15	40.0±8.2	0.02±0.04
	LS	25.5±10.1	0.17±0.18	34.0±13.1	0.12±0.29
	BANANAS	3.1±1.8	0.51±0.21	4.0±2.1	0.77±0.63
	TPE	51.5±17.6	0.07±0.15	**70.5±14.3**	0.02±0.05
	ZAPS	**55.7±3.1**	**0.01±0.05**	67.7±2.5	**< 0.01**

4.4 Ablation Study

We evaluate the individual contribution of each component of ZAPS via ablation.

PROXYFIT ablation. Replacing PROXYFIT with all 13 proxies reduces P@100 and increases Regret on CIFAR-10 and CIFAR-100 (Figure 4). Redundant proxies such as `flops` and `params` provide overlapping information that dilutes the surrogate signal and increases dimensionality without benefit.

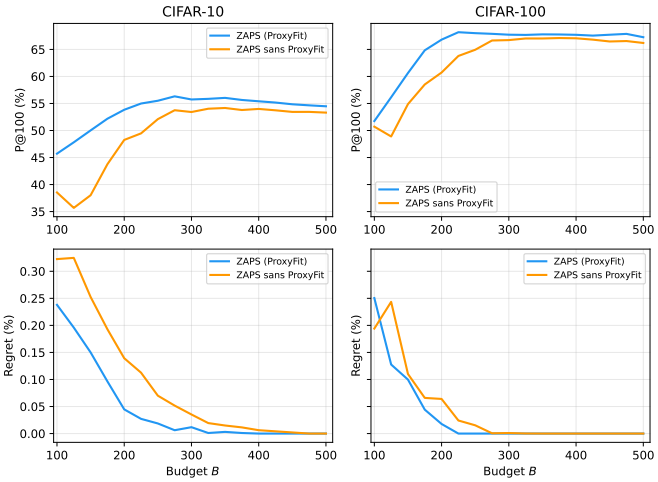

Figure 4: Anti-redundant selection vs. full proxy set.

K-means ablation. The quality of the initial labeled set directly conditions the surrogate's ability to model the search space. A purely random initialization does not guarantee representative coverage: it may over-represent dense regions while ignoring promising areas. The hybrid K-means initialization, by combining exploitation and exploration, provides a more informative initial set, translating into better performance across all datasets (Figure 5).

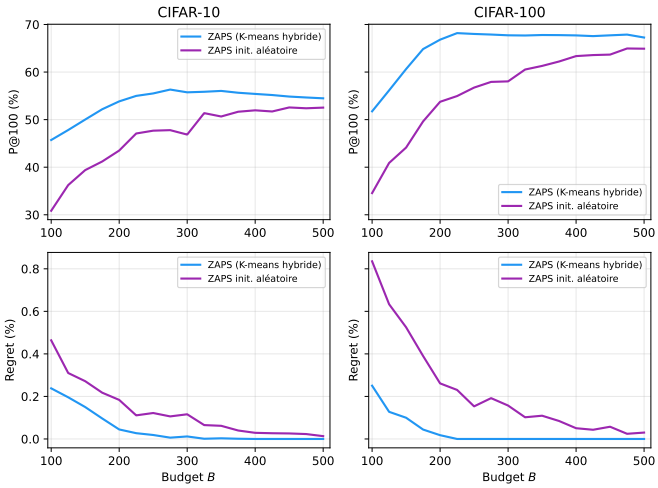

Figure 5: Hybrid K-means initialization vs. random initialization.

Table 3 quantifies the contribution of each component. K-means initialization has the largest impact, contributing $+10.3$ pp on CIFAR-10 and $+13.1$ pp on CIFAR-100 at $B = 200$. PROXYFIT adds $+5.6$ pp and $+6.1$ pp respectively, while simultaneously reducing variance (std from $15.5 \rightarrow 3.5$ on CIFAR-10).

Table 3: Ablation on NAS-Bench-201 (P@100 and Regret at $B = 200$, 200 seeds).

	C10		C100	
Variant	P@100	Reg.	P@100	Reg.
ZAPS (full)	**53.8±3.5**	**0.04**	**66.8±6.9**	**0.02**
w/o PROXYFIT	48.3±6.3	0.14	60.7±10.0	0.06
random init.	43.5±15.5	0.18	53.8±19.7	0.26

4.5 Ranking Quality: Spearman and Kendall

Beyond P@100 and Regret, we evaluate the overall ranking quality of ZAPS via Spearman correlation ρ and Kendall's τ between predicted and true rankings over all 15,625 NAS-Bench-201 architectures. Figure 6 shows Spearman correlation as a function of budget.

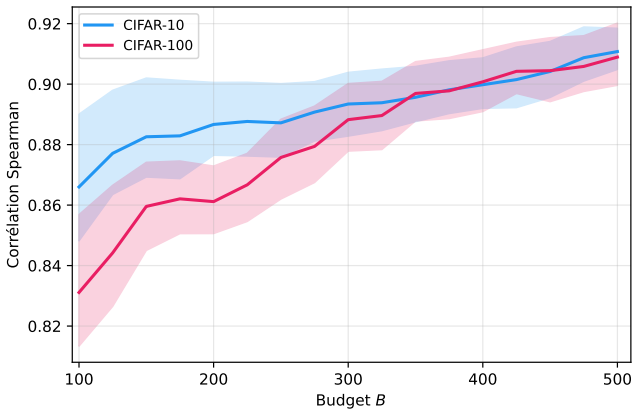

Figure 6: Spearman correlation on NAS-Bench-201 as a function of budget, averaged over 200 seeds.

ZAPS achieves $\rho \geq 0.87$ on CIFAR-10 and $\rho \geq 0.83$ on CIFAR-100 from $B = 100$, and exceeds $\rho = 0.89$ at $B = 200$. In terms of Kendall's τ (estimated via $\tau \approx \frac{2}{\pi} \arcsin \rho$), ZAPS achieves $\tau \geq 0.63$ from $B = 100$ and $\tau \geq 0.64$ at $B = 200$, confirming high ranking quality on both datasets.

4.6 Proxy Transferability

PROXYFIT selects a subset of complementary proxies by computing their Spearman correlation with true accuracy. This raises two practical questions: must PROXYFIT be recalibrated (i) for each new dataset, or (ii) for each new search space?

Cross-dataset transferability. PROXYFIT is calibrated once on CIFAR-10, and the same subset of six proxies is applied directly to CIFAR-100 and ImageNet16-120, without recalibration. The obtained performance is close to that of PROXYFIT recalibrated on each dataset, confirming that complementarity between proxies is a structural property of the search space, independent of the dataset (Figure 7).

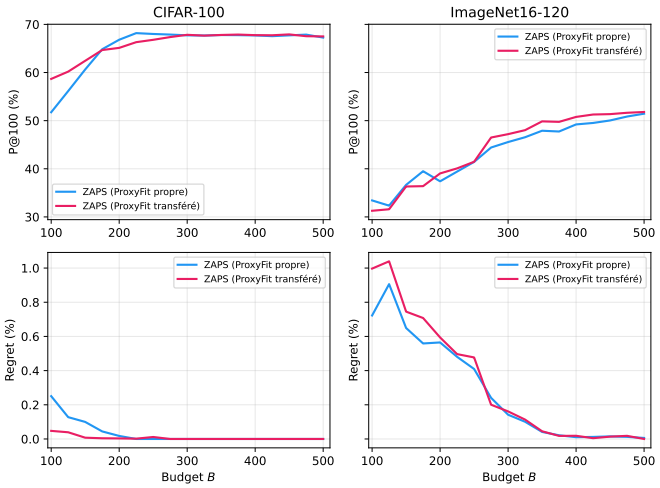

Figure 7: Cross-dataset transferability: subset selected on CIFAR-10 applied to CIFAR-100 and ImageNet16-120, without recalibration.

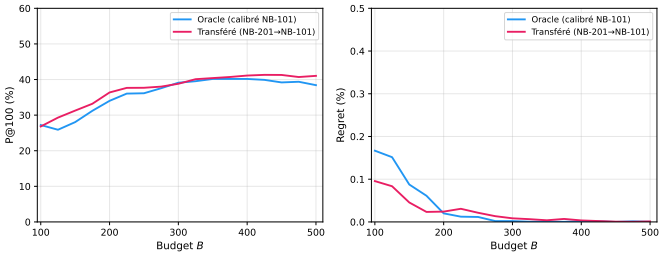

Figure 8: Cross-search-space transferability: ZAPS with PROXYFIT calibrated on NAS-Bench-201 vs. calibrated on NAS-Bench-101, evaluated on NAS-Bench-101/CIFAR-10. The two curves remain close across the full budget range.

Cross-search-space transferability. We examine whether PROXYFIT calibrated on NAS-Bench-201 can be directly transferred to NAS-Bench-101, a different search space belonging to the same family of convolution-based architectures. NAS-Bench-101 contains 423,624 architectures in total; we evaluate ZAPS on the subset of 9,781 architectures annotated by NAS-Bench-Suite-Zero, compared to 15,625 in NAS-Bench-201.

We compare two variants of ZAPS on NAS-Bench-101/CIFAR-10:

- **ZAPS oracle**: PROXYFIT calibrated directly on NAS-Bench-101;
- **ZAPS transferred**: PROXYFIT calibrated on NAS-Bench-201/CIFAR-10 and applied as-is to NAS-Bench-101, without recalibration.

This result indicates that when both search spaces belong to the same architectural family (here, cell-based convolutional networks), the zero-cost proxies and their complementarity hierarchy are preserved across benchmarks. PROXYFIT can therefore be calibrated *once* and reused at no additional cost.

5 Conclusion

We presented ZAPS, a pipeline combining zero-cost proxy selection and surrogate-based active learning for neural architecture search. The central idea is to jointly exploit the complementarity of zero-cost proxies — identified by PROXYFIT — and the topological encodings of architectures to efficiently guide an XGBoost + UCB surrogate. Each component — PROXYFIT, hybrid K-means initialization, bootstrapped MRMR, and XGBoost + UCB — contributes measurably, as confirmed by ablation studies. On NAS-Bench-201, ZAPS achieves **53.8 %** P@100 at $B = 200$ on CIFAR-10, which is $1.5\times$ more than TPE at equivalent budget, with a Spearman correlation $\rho \geq 0.87$ from $B = 100$. ZAPS dominates at moderate budget on CIFAR-10 and CIFAR-100; TPE remains competitive at large budget ($B = 300$) on CIFAR-100, indicating room for improvement at high budgets.

Limitations. ZAPS is validated on tabular benchmarks with discrete search spaces. The one-hot topological encoding is specific to NAS-Bench-201, and PROXYFIT must be re-run for each new search space. Extension to continuous spaces (DARTS) remains an open direction.

Future work. Future directions include adapting ZAPS to larger search spaces, integrating multi-fidelity evaluations as intermediate signals, and developing zero-cost proxies tailored to tasks beyond image classification.

Reproducibility. Code, experiment scripts, and results (200 seeds) will be made publicly available upon publication. Full hyperparameters are listed in Appendix A.

References

M. S. Abdelfattah, A. Mehrotra, Y. Ozkan, and N. D. Lane. Zero-cost proxies for lightweight nas. In *International Conference on Learning Representations*, 2021.

Y. Akhauri, M. S. Abdelfattah, N. D. Lane, and R. Sama. EZNAS: Evolving zero-cost proxies for neural architecture scoring. In *Advances in Neural Information Processing Systems*, 2022.

J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, volume 24, 2011.

T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

P. Dong et al. LPZero: Language model zero-cost proxy search from zero. *arXiv preprint arXiv:2410.04808*, 2024.

X. Dong and Y. Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations*, 2020.

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

N. Lee, T. Ajanthan, and P. H. Torr. Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*, 2019.

G. Li et al. Zico: Zero-shot nas via inverse coefficient of variation on gradients. In *International Conference on Learning Representations*, 2023.

H. Liu, K. Simonyan, and Y. Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

J. Mellor, J. Turner, A. Storkey, and E. J. Crowley. Neural architecture search without training. In *International Conference on Machine Learning*, pages 7588–7598. PMLR, 2021.

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

B. Ru, X. Wan, X. Dong, and J. Roberts. Neural architecture search with bayesian optimisation and optimal transport. In *Advances in Neural Information Processing Systems*, 2021.

M. Shen et al. Active learning for deep neural network architecture search. In *arXiv preprint arXiv:2109.03718*, 2021.

H. Tanaka, D. Kunin, D. L. Yamins, and S. Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In *Advances in Neural Information Processing Systems*, volume 33, pages 6377–6389, 2020.

C. White, W. Nolen, and Y. Savani. Bananas: Bayesian optimization with neural architectures for neural architecture search. In *AAAI Conference on Artificial Intelligence*, 2021.

A. Zela, J. Siems, L. Zimmer, J. Lukasik, M. Keuper, and F. Hutter. NAS-Bench-Suite-Zero: Accelerating research on zero cost proxies. In *Advances in Neural Information Processing Systems*, 2022.

H. Zhu et al. Superproxy: A unified zero-cost proxy for neural architecture search. *arXiv preprint*, 2022.

B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017.

A Hyperparameters

Table 4 lists all hyperparameters of ZAPS used in the experiments. The same values were used across all benchmarks and datasets.

Table 4: Hyperparameters of ZAPS.

Component	Hyperparameter	Value
ProxyFit	Redundancy threshold τ	0.85
	Target size K	6
K-means	Initialization budget n_0	$\lfloor B/4 \rfloor$
	Biased fraction	top 30 % by proxies
	Number of clusters	$n_0/2$
MRMR	Bootstrap resamplings	3
	Minimum subset size	4
	Maximum subset size	10
XGBoost	Number of estimators	50
	Bootstrap replicates	3
	Maximum depth	5
UCB	Exploration coefficient β	0.5
Active loop	Active iterations T	2
	Batch size b	$\lfloor (B - n_0)/T \rfloor$

B Budget Evolution Curves

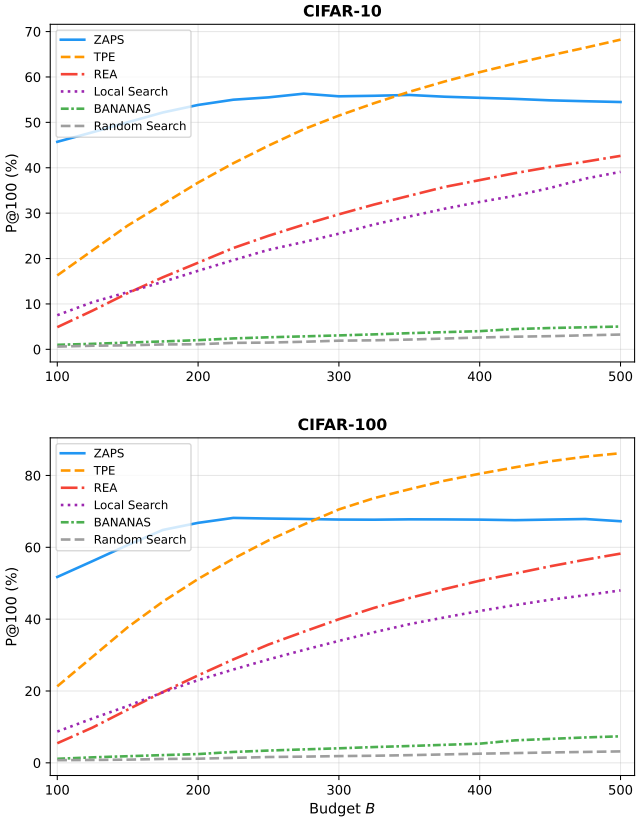

Figure 9: P@100 (%) as a function of budget B.

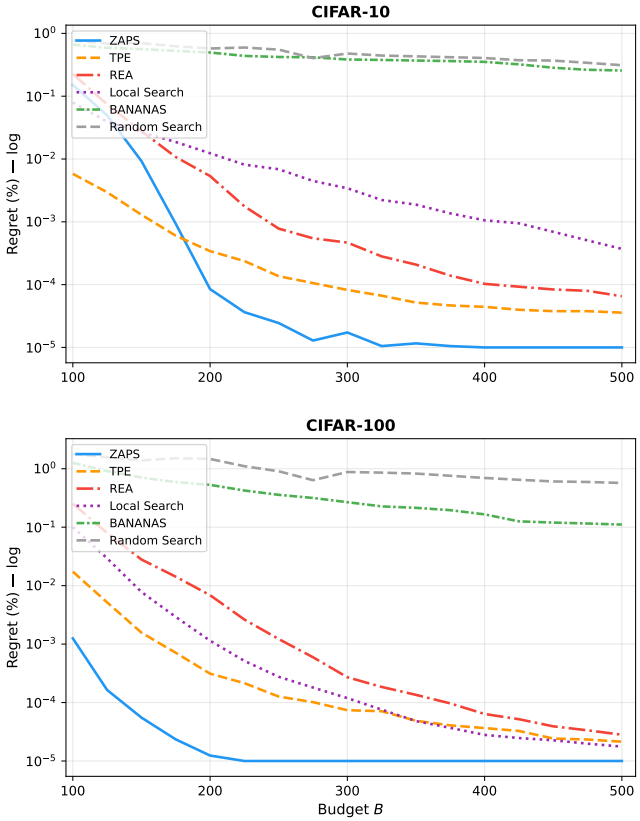

Figure 10: Regret (%) as a function of budget B (log scale).