ZAPS: ZERO-COST ACTIVE PROXY SEARCH FOR NEURAL ARCHITECTURE SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural Architecture Search (NAS) automates network design, but evaluating a single candidate requires training it to convergence, making exhaustive search intractable. Zero-cost proxies estimate architecture quality at initialization in seconds, yet a single proxy is noisy, and combining several does not straightforwardly help: proxies are strongly correlated, so naive aggregation compounds their shared errors instead of averaging them out. Existing methods exploit either proxy signals or architectural topology—never both within a single active-learning framework. We introduce ZAPS (*Zero-cost Active Proxy Search*), a four-stage pipeline that closes this gap. ZAPS (i) selects a compact, non-redundant proxy subset offline via PROXYFIT, a greedy anti-redundancy criterion; (ii) seeds the search with a hybrid K-means strategy that balances exploitation and exploration; (iii) re-selects proxies at every iteration by a bootstrapped vote as the labeled set grows; and (iv) ranks candidates with an XGBoost ensemble trained jointly on proxy ranks and one-hot topological encodings, queried through an Upper Confidence Bound (UCB) acquisition function. On NAS-Bench-201 under a budget of $B = 200$ evaluations, ZAPS recovers **52.3 %** of the true top-100 architectures on CIFAR-10 and **65.8 %** on CIFAR-100, ahead of every baseline we consider—Random Search, Local Search, REA, BANANAS and TPE—and, on CIFAR-10, with less than half the run-to-run standard deviation of the strongest of them. The advantage is largest where evaluations are scarce: on NAS-Bench-201 it narrows as the budget grows, whereas on the harder NAS-Bench-101, which no method comes close to saturating, it widens instead. All methods are scored by a single criterion: how much of the true top-100 lies among the architectures they actually evaluated.

1 INTRODUCTION

Neural Architecture Search (NAS) automates the design of high-performing networks by searching a predefined space of candidate architectures (Elsken et al., 2019). Its dominant cost is evaluation: obtaining the accuracy of a single candidate requires training it to convergence, so covering a realistic search space demands thousands of GPU hours. Reducing the number of such evaluations—or replacing them with cheaper signals—is therefore the central problem in practical NAS.

Zero-cost proxies and their limits. Zero-cost proxies (Mellor et al., 2021; Tanaka et al., 2020) score an architecture at initialization from a single forward or backward pass, cutting evaluation time from hours to seconds. Each proxy probes a distinct network property—expressivity, gradient flow, parameter count—which suggests combining several of them to obtain a broader signal. In practice, however, proxies are far from independent: on NAS-Bench-201 / CIFAR-10, `flops` and `params` correlate at $\rho = 0.995$. Combining proxies naively therefore duplicates information and amplifies correlated noise rather than enriching the signal.

The gap. Existing methods fall into two disjoint families. *Proxy-based* methods rely on a single score (Mellor et al., 2021; Tanaka et al., 2020; Li et al., 2023) or aggregate several through a uniform vote (Abdelfattah et al., 2021); they neither filter redundancy nor exploit the topological structure of the architecture. *Surrogate-based* methods (White et al., 2021; Ru et al., 2021) learn a predictor over architecture encodings but ignore zero-cost signals entirely, and consequently need substantially larger labeled sets before their surrogate becomes accurate. To our knowledge, no prior work simultaneously

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

(a) selects a non-redundant, high-relevance proxy subset and (b) fuses it with architectural topology inside an active-learning loop.

Our approach. We propose ZAPS, which is built on the hypothesis that proxy scores and architectural structure are *complementary* rather than interchangeable: proxies measure functional quality at initialization, whereas a one-hot topology encoding describes the computation graph that determines expressive capacity. ZAPS feeds both into a single surrogate and queries it through a UCB acquisition function, so that every evaluation spent is chosen to be maximally informative. Our ablation (Section 4.4) confirms that each pipeline component is necessary: removing the anti-redundancy filter costs 11–13 pp of P@100, and removing the structured initialization costs 16–23 pp while multiplying the run-to-run standard deviation by two to three.

Contributions.

1. **ProxyFit** (Section 3.1): an offline greedy algorithm that extracts a compact proxy subset maximizing correlation with ground-truth accuracy under an explicit anti-redundancy constraint.
2. **Hybrid K-means initialization** (Section 3.2): a two-phase seeding strategy that balances exploitation of proxy-favored regions against coverage of the remaining search space.
3. **Bootstrapped online re-selection** (Section 3.3): a dynamic rule that adapts the active proxy subset to the growing labeled set, stabilized by a bootstrap majority vote and halted by an MRMR stopping criterion.
4. **A proxy-augmented surrogate** (Section 3.4): an XGBoost ensemble trained on concatenated proxy ranks and one-hot topology, driving a UCB acquisition function.
5. **An empirical study** (Section 4) over 200 independent seeds on two search spaces, NAS-Bench-201 and NAS-Bench-101, establishing that ZAPS leads every baseline we consider at moderate budget while being markedly more stable across seeds. Every baseline is audited against its source publication and a behavioral check before use.

2 Related Work

Zero-cost proxies. NASWOT (Mellor et al., 2021) scores architectures by the log-determinant of a kernel matrix built from the binary activation patterns of their ReLU units, which measures how distinctly an untrained network separates inputs. Synflow (Tanaka et al., 2020) propagates a data-independent gradient signal that avoids layer collapse. SNIP (Lee et al., 2019) measures connection saliency, and ZiCo (Li et al., 2023) quantifies gradient stability across layers. Despite their efficiency, each of these scores captures a single facet of architectural quality and its predictive power varies considerably across search spaces and datasets. NAS-Bench-Suite-Zero (Zela et al., 2022) standardizes this landscape by releasing precomputed values for 13 proxies over several benchmarks; we build directly on this resource.

Proxy combination. Abdelfattah et al. (2021) ensemble zero-cost proxies by a majority vote among three fixed proxies, weighting each equally and without discounting the redundancy between them. EZNAS (Akhauri et al., 2022) evolves symbolic composite operators over proxy statistics, and LPZero (Dong et al., 2024) transfers this idea to language-model search spaces. UP-NAS (Huang et al., 2024) learns a multi-proxy estimator that consolidates several zero-cost proxies into a unified score, but the estimator itself must be trained on labeled architectures. All of these produce a *single scalar score*; none feeds the selected proxies into a learned surrogate alongside structural features. ProxyFit differs on both counts: it selects proxies for joint relevance and mutual complementarity, and passes the resulting subset to a surrogate that also consumes architectural topology.

Surrogate-based NAS. BANANAS (White et al., 2021) represents architectures by path encodings and fits an ensemble surrogate for Bayesian optimization; NASBOWL (Ru et al., 2021) employs a graph-kernel Gaussian process at higher computational cost. Both learn exclusively from architecture encodings and require comparatively large labeled sets before the surrogate becomes reliable—a regime our budgets deliberately exclude. ZAPS augments this family by injecting zero-cost proxy

ranks into the surrogate's feature space; to our knowledge it is the first method to combine the two signal types in a single surrogate under a strict evaluation budget.

Active learning and classical NAS. Active-learning approaches to NAS (Ma et al., 2019) choose which architectures to label next based on surrogate uncertainty, again using encodings alone. Classical NAS relies on reinforcement learning (Zoph & Le, 2017) or regularized evolution (Real et al., 2019), at a cost of thousands of GPU days; DARTS (Liu et al., 2019) reduces this through a differentiable relaxation but does not apply to the tabular benchmarks studied here. ZAPS combines UCB-driven active learning with a proxy-augmented surrogate, inheriting the sample efficiency of the former and the cheap prior of the latter.

3 METHOD

Problem setting. Let $\mathcal{A}$ be a discrete architecture search space and let $f : \mathcal{A} \to \mathbb{R}$ denote true validation accuracy, accessible only through full training. Given a budget B of such evaluations, a method produces a set $\mathcal{E}_B \subseteq \mathcal{A}$ of B architectures it chose to evaluate. Our primary objective is to recover $\mathcal{T}_{100}$, the 100 best architectures in $\mathcal{A}$, and we measure how much of it a method actually reached:

$$\text{P@100} \;=\; \tfrac{1}{100}\,\big|\,\mathcal{E}_B \cap \mathcal{T}_{100}\,\big| \;\times\; 100\,\%. \tag{1}$$

Scoring on $\mathcal{E}_B$ rather than on a predicted ranking is what makes the comparison meaningful: search methods such as Random Search or REA produce no ranking over $\mathcal{A}$, so a predicted top-100 is undefined for them, and crediting surrogate methods for architectures they never trained would compare unlike quantities. Every method here is scored by Eq. 1. As a secondary criterion we report *regret*, $R = (f^\star - f_{\text{best}})/f^\star \times 100\,\%$, where $f^\star = \max_{a\in\mathcal{A}} f(a)$ and $f_{\text{best}} = \max_{a\in\mathcal{E}_B} f(a)$. P@100 thus measures how much of the top region a method reaches; regret measures how good its single best find is.

Pipeline overview. ZAPS comprises four stages (Figure 1): **(1)** offline proxy selection with PROXYFIT; **(2)** hybrid K-means initialization of the labeled set; **(3)** dynamic proxy re-selection by a bootstrapped vote; and **(4)** UCB-guided querying of an XGBoost ensemble. Stage 1 runs once; stages 3 and 4 alternate inside the active-learning loop (Algorithm 3).

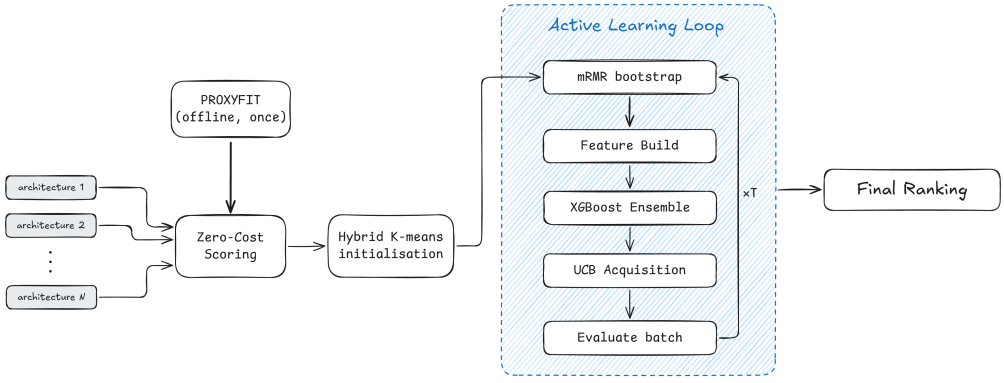


Figure 1: The four stages of ZAPS. PROXYFIT runs once offline and returns a proxy subset; the hybrid K-means seeds the labeled set; re-selection, surrogate fitting and UCB querying then alternate for T iterations, each one spending part of the budget on newly evaluated architectures.

3.1 PROXYFIT: ANTI-REDUNDANT PROXY SELECTION

Let P denote the pool of $|P| = 13$ available zero-cost proxies (Table 7). Supplying all of them to the surrogate injects correlated noise and dilutes the informative directions of the feature space. PROXYFIT instead extracts a compact subset $\Pi \subseteq P$ of size K: it traverses the proxies in decreasing order of $|\rho(p, f)|$—their Spearman correlation with accuracy—and admits a candidate only when its correlation with every already-selected proxy stays below a threshold τ (Algorithm 1). Relevance thus determines the traversal order, while redundancy acts as an admission filter.

On CIFAR-10 with $\tau = 0.85$ and $K = 6$, PROXYFIT returns {nwot, jacov, synflow, params, snip, grasp}. Notably it rejects flops, whose correlation with the already-admitted params reaches 0.995; Appendix G shows the full correlation structure that drives these decisions.

On the use of ground-truth accuracies. PROXYFIT computes $\rho(p, f)$ over the whole space, so as stated it consumes labels that a search method operating under budget B would not have. We make this explicit rather than leave it implicit, and we neutralize it in two ways. First, PROXYFIT is an *offline, one-off* calibration whose output is a list of six proxy names—not a ranking of architectures—so it is reusable across every subsequent search on that space. Second, and decisively, the calibration turns out not to matter: substituting a subset calibrated on a *different* dataset leaves performance unchanged within 5.2 pp of P@100 at $B \geq 200$ (Section 4.6). The reported gains therefore do not stem from privileged access to target labels, and the headline conclusions of Section 4.2 hold with a transferred subset.

3.2 HYBRID K-MEANS INITIALIZATION

Surrogate quality under a tight budget depends critically on the initial labeled set $\mathcal{L}_0$. Uniform random sampling wastes evaluations on uninformative regions, whereas selecting purely by proxy score confines the surrogate to a narrow, proxy-biased slice of the space and leaves it unable to correct proxy errors. We therefore allocate the initialization budget n_0 across two complementary phases:

- **Exploitation phase** ($n_0/2$ architectures): K-means restricted to the top 30 % of $\mathcal{A}$ ranked by aggregate proxy score, retaining the architecture closest to each centroid. This concentrates labels where high-accuracy architectures are likely to lie, while clustering prevents the sample from collapsing onto a single mode.
- **Coverage phase** ($n_0/2$ architectures): K-means over the entire space minus the points already chosen, again retaining one representative per cluster. Because this pass is unrestricted, it reaches the regions the proxies rank poorly—precisely where the surrogate must learn to correct them—while its own clustering keeps the sample spread out.

The two phases are thus not disjoint: the second clusters over the whole space minus the points already chosen, so it ranges over the proxy-favored region as well rather than being confined to its complement. The exploitation phase acts as a deliberate over-sampling of that region rather than as a partition of the space. Both phases cluster in proxy-rank space. We set $n_0 = \max(10, \lfloor B/4 \rfloor)$ (50 architectures at $B = 200$), leaving the remainder for the active loop; when $B - n_0$ is not divisible by T, the floor leaves at most $T - 1$ evaluations of the budget unspent.

3.3 BOOTSTRAPPED ONLINE RE-SELECTION

The subset chosen offline by PROXYFIT is calibrated on the search space as a whole, but the relevance and redundancy of a proxy *conditioned on the labeled set* shift as $\mathcal{L}_t$ grows. We therefore re-select an active subset Π_t at every iteration, applying the same relevance-ordered, redundancy-filtered traversal as PROXYFIT but with all correlations re-estimated on $\mathcal{L}_t$ alone.

Two elements differ from the offline stage. First, we add a stopping rule based on the Maximum Relevance Minimum Redundancy score of a candidate p against the partial selection S built so far,

$$J(p \mid S) \;=\; |\rho(p, f)| \;-\; \frac{1}{|S|} \sum_{q \in S} |\rho(p, q)|, \tag{2}$$

which halts the traversal once the leading candidate contributes more redundancy than relevance ($J(p \mid S) < 0$), provided $|S| \geq k_{\min}$. Second, because correlations estimated on a small $\mathcal{L}_t$ are unstable, we repeat the traversal on $B_{\text{boot}} = 3$ bootstrap resamples and keep the proxies with the most votes (Algorithm 2). The target size k grows with the labeled set up to a ceiling, so that model capacity tracks the available evidence.

3.4 PROXY-AUGMENTED SURROGATE AND UCB ACQUISITION

Feature representation. Each architecture a is represented by $\phi(a) = \big[\hat{\mathbf{r}}(a);\, \mathbf{e}(a)\big]$, the concatenation of *(i)* normalized proxy ranks $\hat{\mathbf{r}}(a) \in [0, 1]^{|\Pi_t|}$, computed over the whole space so that a

single outlying proxy value cannot dominate the feature, and *(ii)* a binary encoding $\mathbf{e}(a)$ of the cell's structure: on NAS-Bench-201, one-hot over the operation assigned to each of the six cell edges ($6 \times 5 = 30$ dimensions); on NAS-Bench-101, whose operations sit on nodes rather than edges, the strict upper triangle of the 7×7 adjacency matrix (21 bits) concatenated with a one-hot operation code per node ($7 \times 5 = 35$), for 56 in total. The first block carries a cheap functional prior available before any training; the second carries the exact discrete structure the proxies can only summarize. Their concatenation is what distinguishes ZAPS from both proxy-only and encoding-only methods; to our knowledge no prior zero-cost method supplies both signal types to a single regressor.

Ensemble and acquisition. We fit three XGBoost (Chen & Guestrin, 2016) regressors on independent bootstrap resamples of $\mathcal{L}_t$, yielding for every unevaluated architecture a predictive mean $\mu(a)$ and a dispersion $\sigma(a)$ that serves as an uncertainty estimate. The next batch of $b = \lfloor (B - n_0)/T \rfloor$ architectures maximizes the Upper Confidence Bound

$$\alpha(a) \;=\; \mu(a) \;+\; \beta\,\sigma(a), \qquad \beta = 0.5, \tag{3}$$

balancing exploitation of high predicted accuracy against exploration of regions where the ensemble disagrees. The loop repeats for T iterations until the budget is exhausted, re-selecting proxies and refitting the ensemble each time; Algorithm 3 in Appendix B states the assembled procedure.

4 Experiments

4.1 Experimental Setup

Benchmarks. Our main evaluation uses NAS-Bench-201 (Dong & Yang, 2020), a tabular benchmark enumerating 15,625 architectures generated by assigning one of five operations to each of six cell edges, with recorded accuracies on CIFAR-10 and CIFAR-100. Because every accuracy is a table lookup, no network is trained at any point and results are exactly reproducible. Section 4.5 repeats the comparison on NAS-Bench-101 (Ying et al., 2019), and the transferability study (Section 4.6) additionally uses ImageNet16-120. Proxy values are taken unmodified from NAS-Bench-Suite-Zero (Zela et al., 2022). A single ZAPS run at $B = 200$ completes in roughly five seconds on CPU.

Proxies. We use all 13 proxies released by NAS-Bench-Suite-Zero, spanning deterministic (`params`, `flops`, `l2_norm`, `synflow`), activation (`nwot`, `jacov`, `epe_nas`, `zen`, `plain`) and gradient (`fisher`, `snip`, `grad_norm`, `grasp`) families. Individually their Spearman correlation with accuracy on NAS-Bench-201 / CIFAR-10 ranges from -0.27 to 0.77; Table 7 in Appendix G gives the breakdown.

Metrics. We report P@100 (Eq. 1), the percentage of the true top-100 architectures a method actually evaluated within its budget (higher is better), and regret, the relative accuracy gap between the global optimum and the best architecture it evaluated (lower is better). As a complementary diagnostic specific to ZAPS, we report the Spearman correlation between its surrogate's predicted ranking and the ground-truth ranking over all 15,625 architectures; this quantity is defined only for methods that rank the whole space and is therefore not used for comparison. Unless stated otherwise, every number is a mean over 200 independent random seeds.

Baselines. We compare against two families. *Search algorithms*: Random Search (RS), Local Search (LS), regularized evolution (REA) (Real et al., 2019), BANANAS (White et al., 2021), and TPE (Bergstra et al., 2011), each granted the same budget B of true evaluations and run over the same 200 seeds. *Zero-cost proxy combinations* (Section 4.3): the best single proxy, `synflow` used without calibration, the rank product, and a Borda aggregation over the three proxies recommended by Abdelfattah et al. (2021). These rank the space from proxy scores alone; to spend the same budget as the search algorithms, each then trains the top B architectures of its own ranking.

4.2 Main Results

Table 1 reports performance at $B = 200$ and $B = 300$; Figure 2 traces the full budget range.

At $B = 200$, ZAPS recovers $52.3\,\%$ of the true top-100 on CIFAR-10, which is $1.5\times$ the score of the strongest baseline, TPE ($35.4\,\%$), and 16.9 pp above it. The margin is comparable on CIFAR-100, where ZAPS reaches $65.8\,\%$ against $50.5\,\%$ for TPE.

Beyond the mean, ZAPS is also the most *stable* method: at $B = 200$ its standard deviation is 6.1 pp on CIFAR-10 against 15.0 for TPE and 10.6 for BANANAS, and 11.0 pp on CIFAR-100 against 16.9 and 11.7. Under a single-shot budget, where a practitioner cannot average over seeds, this matters as much as the mean.

Behaviour across budget regimes. The advantage of ZAPS is largest where evaluations are scarce and narrows as they accumulate (Figure 2). On CIFAR-10 it is stable, from $+16.9$ pp over the best baseline at $B = 200$ to $+13.8$ pp at $B = 500$. On CIFAR-100 it decays markedly, from $+15.4$ pp at $B = 200$ to a negligible $+0.6$ pp at $B = 500$, where BANANAS, TPE and REA have nearly caught up with ZAPS. This is what the design predicts: the zero-cost prior is informative from the first iteration, but its marginal value diminishes once enough true labels accumulate for a surrogate to be fitted without it (Appendix I gives the head-to-head curves).

Statistical significance. With 200 seeds per cell we test these gaps rather than assert them, using a one-sided Mann–Whitney U test on per-seed P@100 values and Cliff's δ as effect size ($|\delta| < 0.15$ negligible, above 0.47 large). At $B = 200$ the advantage is unambiguous on both datasets ($p < 10^{-24}$ against every baseline), with a large effect against the strongest of them ($\delta = 0.69$ on CIFAR-10, 0.60 on CIFAR-100, both versus TPE), and remains large at $B = 300$ on CIFAR-10 ($p < 10^{-22}$, and $\delta = 0.57$ versus TPE, the weakest effect). On CIFAR-100 at $B = 300$, however, the gap to TPE is still detectable ($p < 0.01$) while δ has fallen to a negligible 0.14; against BANANAS, the best on average, it remains medium ($\delta = 0.45$): detectability and practical relevance part company, so we report both rather than p alone. On regret the tests are less favourable—ZAPS does not separate from BANANAS on CIFAR-10 at $B = 200$ ($p = 0.25$), a point Section 5 takes up.

Baseline implementations. Every baseline is our own re-implementation, since no reference code targets these benchmarks in a directly usable form. Because a comparison is only as meaningful as its weakest baseline, we audited each against its source publication and against a behavioral check—BANANAS reproduces the global CIFAR-10 optimum by $B = 500$ as in the original work, and Random Search matches its hypergeometric expectation to within 0.2 pp—and we report those checks rather than asking the reader to take faithfulness on trust. Two conventions affect comparability and are stated explicitly in Appendix A: the budget counts *distinct* architectures, and the projection of a mutated encoding back onto NAS-Bench-101 must exclude the parent, without which a nearest-neighbour projection would reduce REA to random sampling.

Ranking quality. The surrogate is accurate globally, not only near the optimum: from $B = 100$ onward it attains a Spearman correlation of $\rho \geq 0.86$ on CIFAR-10 and $\rho \geq 0.83$ on CIFAR-100 against the full ground-truth ranking, rising to $\rho = 0.89$ at $B = 200$ (Appendix H).

4.3 Comparison with Zero-Cost Proxy Combination

Since our premise is that proxies must be *selected* rather than merely aggregated, we compare against the proxy-combination schemes ProxyFit is designed to improve upon. A zero-cost method produces its ranking without training anything, so that ranking cannot be set directly against a method that has spent B evaluations. We therefore give each scheme the budget it would consume in practice: it ranks the space from proxy scores, then trains the top B of that ranking. Every row of Table 6 in Appendix E thus spends exactly $B = 200$ true evaluations, and P@100 counts for all of them how much of the true top-100 was actually trained. The proxy schemes are deterministic, so no standard deviation is reported.

The strongest scheme, `synflow` used alone, recovers $22.0\,\%$ of the true top-100 on CIFAR-10 and $36.0\,\%$ on CIFAR-100 for the same 200 evaluations. That places it above Random Search and Local Search on both datasets, and above REA on CIFAR-100, but below the surrogate-based baselines BANANAS and TPE (Table 1): a well-chosen proxy is a strong starting point, not a substitute for search. Two further observations bear on the motivation for ProxyFit, and neither involves ZAPS.

Table 1: Comparison on NAS-Bench-201 at $B = 200$ and $B = 300$ evaluations. All methods are scored identically: P@100 counts the true top-100 architectures among those the method actually evaluated. Values are mean $\pm$ standard deviation over 200 independent seeds. Best value per column and budget in **bold**. Every P@100 gap between ZAPS and every baseline is significant at $p < 0.01$ (one-sided Mann–Whitney U); effect sizes and the one case where significance does not imply practical relevance are discussed in the text.

		CIFAR-10		CIFAR-100	
B	Method	P@100 (%)	Regret (%)	P@100 (%)	Regret (%)
200	RS	1.4 ± 1.1	0.65 ± 0.26	1.4 ± 1.1	1.66 ± 0.87
	LS	17.3 ± 9.1	0.25 ± 0.22	23.0 ± 13.1	0.30 ± 0.51
	REA	27.0 ± 7.4	0.11 ± 0.15	35.1 ± 9.7	0.05 ± 0.10
	BANANAS	34.7 ± 10.6	0.06 ± 0.12	48.0 ± 11.7	0.03 ± 0.05
	TPE	35.4 ± 15.0	0.12 ± 0.18	50.5 ± 16.9	0.06 ± 0.19
	ZAPS	$\mathbf{52.3} \pm 6.1$	$\mathbf{0.04} \pm 0.09$	$\mathbf{65.8} \pm 11.0$	$\mathbf{0.02} \pm 0.14$
300	RS	2.0 ± 1.4	0.59 ± 0.25	1.9 ± 1.4	1.36 ± 0.79
	LS	25.5 ± 10.1	0.17 ± 0.18	34.0 ± 13.1	0.11 ± 0.29
	REA	43.6 ± 8.8	0.05 ± 0.13	59.1 ± 7.6	$\mathbf{0.00} \pm 0.01$
	BANANAS	51.3 ± 7.5	$\mathbf{0.01} \pm 0.04$	70.2 ± 8.8	0.01 ± 0.04
	TPE	49.4 ± 18.3	0.09 ± 0.17	69.5 ± 16.0	0.03 ± 0.14
	ZAPS	$\mathbf{65.1} \pm 5.8$	0.01 ± 0.05	$\mathbf{76.1} \pm 4.5$	$\mathbf{0.00} \pm 0.00$

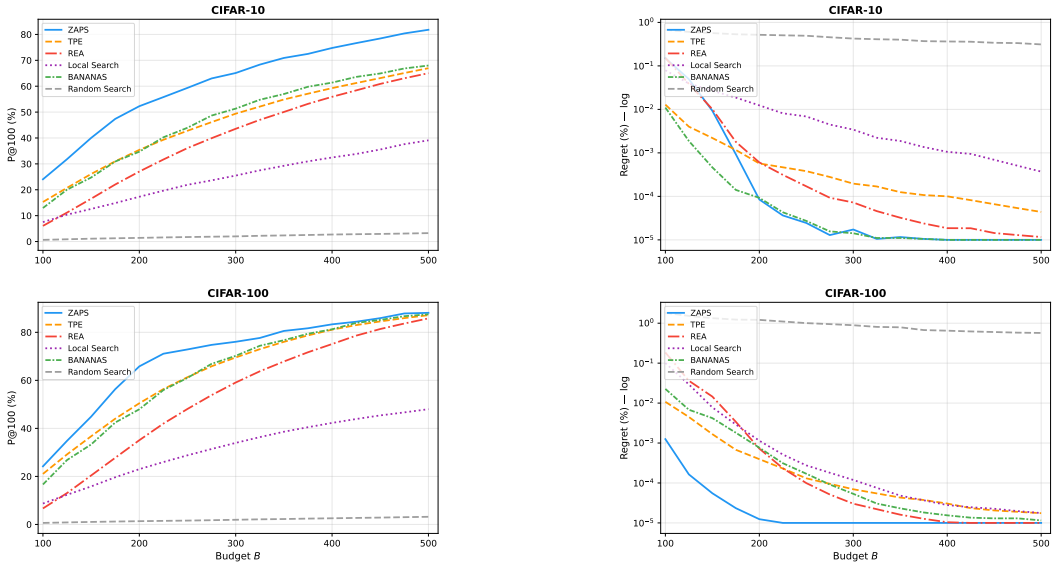


Figure 2: P@100 (left) and regret (right) against the evaluation budget B on NAS-Bench-201, over 200 seeds; ZAPS separates from every baseline early. P@100 curves are means; regret curves are geometric means of the per-seed regret floored at 10^{-5}, so a curve resting on the floor marks budgets at which every seed reaches the global optimum.

Naive combination is worse than no combination. Every aggregation scheme falls far short of `synflow` used alone: on CIFAR-10 the rank product over all thirteen proxies and the Borda aggregation over the three proxies of Abdelfattah et al. (2021) both reach $0.0\,\%$ against $22.0\,\%$ for `synflow`, and restricting the product to positively correlated proxies does not rescue it. Selecting the single proxy by Spearman ρ fares no better—it picks `nwot` ($\rho = 0.775$, the highest of the thirteen), which also reaches $0.0\,\%$: a proxy can order the bulk of the space well while being uninformative exactly where a NAS method operates, which is why we report P@100 rather than the Spearman correlation common in the zero-cost literature. Aggregating without accounting for redundancy destroys the signal each proxy carried on its own, because correlated proxies reinforce one another's errors instead of averaging them out—the failure mode ProxyFit is built to avoid, and which Section 4.4 measures again inside the full pipeline.

4.4 Ablation Study

We isolate the contribution of the two components that shape the surrogate's input: the anti-redundant proxy selection of ProxyFit and the structured initialization. Table 2 reports both at $B = 200$; Appendix D shows the effect across the budget range.

Structured initialization is the dominant factor. Replacing hybrid K-means by uniform random sampling costs 15.6 pp of P@100 on CIFAR-10 and 22.6 pp on CIFAR-100. More strikingly, it inflates the standard deviation from 6.1 to 17.3 pp on CIFAR-10 and quadruples regret ($0.045 \rightarrow 0.181$). Without a structured seed the outcome of a run is largely decided by whether the initial sample happened to land near a good region—exactly the fragility a small budget cannot absorb. Note that this variant still uses the ProxyFit subset, so the figure isolates the seeding strategy alone.

Anti-redundancy filtering contributes consistently. Seeding the hybrid K-means with all thirteen proxies instead of the ProxyFit subset costs 11.4 pp on CIFAR-10 and 12.8 pp on CIFAR-100, and raises the standard deviation from 6.1 to 10.8 pp. Only the initialization differs here: in both cases the surrogate is fed the subset returned by the online re-selection, so the comparison isolates ProxyFit itself. Redundant proxies distort the aggregate score that defines the exploitation region, so the initial labels concentrate where a signal counted several times over happens to point. Regret follows, tripling from 0.045 % to 0.139 % on CIFAR-10. This is the same failure mode observed in Section 4.3, where naive rank aggregation underperformed the single best proxy—here measured inside the full pipeline rather than on the raw proxy rankings.

Table 2: Component ablation on NAS-Bench-201 at $B = 200$, over 200 seeds. Each row removes a single component from the full pipeline. Both variants alter only the initialization—the first seeds the hybrid K-means with all thirteen proxies, the second replaces it by uniform sampling—while the online re-selection and the surrogate are unchanged.

	CIFAR-10		**CIFAR-100**	
Variant	P@100 (%)	Regret (%)	P@100 (%)	Regret (%)
ZAPS (full pipeline)	**52.3** $\pm$ 6.1	**0.045**	**65.8** $\pm$ 11.0	**0.018**
w/o ProxyFit (init. on all 13)	40.8 $\pm$ 10.8	0.139	53.0 $\pm$ 15.4	0.064
w/o K-means (random init.)	36.7 $\pm$ 17.3	0.181	43.2 $\pm$ 21.4	0.261

Do the two feature blocks actually complement each other? Our central hypothesis is that proxy ranks and topological encodings carry different information. We test it directly by cutting the surrogate's input in two, leaving the rest of the pipeline untouched so that any difference is attributable to the features alone: *proxies only* keeps the re-selected proxy ranks, *topology only* keeps the 30-dimensional one-hot encoding (Table 3; curves in Appendix F).

Both blocks contribute significantly, but far from symmetrically. Removing the topology costs 4.5 pp on CIFAR-10 and 7.1 pp on CIFAR-100 ($p < 10^{-7}$); removing the proxies costs 21.2 and 22.4 pp ($p < 10^{-53}$). The proxies are the dominant signal, which is consistent with the rest of the pipeline, since ProxyFit and the initialization both operate on proxy ranks.

Two observations qualify this. First, the topology-only variant reaches 31.1 % on CIFAR-10—*below* BANANAS (34.7 %), itself a surrogate over architecture encodings. Stripped of its proxies, our pipeline is an ordinary encoding-based surrogate; the proxies are what lift it to 52.3 %. Second, the contribution of the topology shrinks with the budget, from +4.6 pp at $B = 100$ to +1.5 pp at $B = 500$ on CIFAR-10, whereas the gap to the topology-only variant never falls below 15 pp at any budget. A surrogate deprived of proxies never catches up; one deprived of topology largely does, once enough labels accumulate for it to learn the structure directly.

4.5 A Second Search Space: NAS-Bench-101

NAS-Bench-201 is close to saturated: on CIFAR-100, several methods exceed 85 % P@100 at $B = 500$ with regret below 0.1 %. To check our results are not an artefact of an easy benchmark, we repeat the comparison on the harder NAS-Bench-101 (Ying et al., 2019): cells vary in node count and carry an adjacency matrix alongside operations, the space holds 423,624 distinct architectures

Table 3: Fusion ablation on NAS-Bench-201 at $B = 200$, 200 seeds. Only the surrogate's input changes; PROXYFIT, initialization, re-selection and UCB acquisition are identical across rows. The proxy block is the online re-selection's output, whose size grows with the labeled set: 7 proxies at the first fit, 10 thereafter.

Surrogate input	Dim.	**CIFAR-10**	**CIFAR-100**
Proxy ranks + topology	37–40	$\mathbf{52.3} \pm 6.1$	$\mathbf{65.8} \pm 11.0$
proxy ranks only	7–10	47.8 ± 8.4	58.7 ± 14.0
topology only	30	31.1 ± 8.0	43.4 ± 11.0

(we use the 9,781 annotated by NAS-Bench-Suite-Zero), and the encoding is 56-dimensional rather than 30.

At $B = 200$, ZAPS reaches $31.5\,\%$ P@100 against $18.4\,\%$ (TPE), $7.2\,\%$ (REA) and $2.0\,\%$ (Random Search); at $B = 500$: 63.0, 37.0, 20.1 and $5.0\,\%$ (Table 9, Appendix K). Local Search and BANANAS are omitted: both rely on a neighbourhood or mutation operator whose transfer to variable-topology graphs raises design questions we did not want to settle for them.

The benchmark is markedly harder—the strongest baseline, TPE, reaches only $37.0\,\%$ at $B = 500$, against $67.0\,\%$ on NAS-Bench-201 / CIFAR-10—though the ordering is unchanged. More telling is the gap's shape: it *widens* with budget, from $+4.3$ pp at $B = 100$ to $+26.0$ pp at $B = 500$, opposite the narrowing seen on CIFAR-100 (Section 4.2). Where the space is too large for any method to exhaust it, the proxy prior keeps paying off, though the surrogate's global Spearman correlation is correspondingly lower (0.62 at $B = 200$ vs 0.89 on NAS-Bench-201): ZAPS models this space less well overall while still locating its top region efficiently.

4.6 TRANSFERABILITY OF THE SELECTED PROXIES

PROXYFIT requires ground-truth accuracies, so a natural objection is that it must be re-run—at cost—for every new dataset or space. It need not be: applying the CIFAR-10 subset directly to CIFAR-100 and ImageNet16-120 costs essentially nothing. Beyond the smallest budgets, the transferred subset stays close to one calibrated on the target dataset, with neither ahead systematically—on ImageNet16-120 it is even the better of the two at most budgets. At the smallest budget, transfer is clearly ahead on CIFAR-100, the target calibration apparently overfitting its offline estimate. Transferring the NAS-Bench-201 subset to the topologically different NAS-Bench-101 likewise tracks an oracle calibrated directly on it (Appendices J, K).

Proxy redundancy thus appears to be a property of the architectures themselves, not the dataset, so PROXYFIT can be treated as a reusable one-off calibration, removing its cost from any subsequent search's budget and supporting the claim in Section 3.1 that ground-truth access does not drive our results.

5 CONCLUSION

ZAPS unifies zero-cost proxy selection and surrogate-based active learning under a strict evaluation budget. On NAS-Bench-201 at $B = 200$ it recovers $52.3\,\%$ of the true top-100 on CIFAR-10 and $65.8\,\%$ on CIFAR-100, ahead of every baseline by at least 15 pp and, on CIFAR-10, with less than half the strongest's run-to-run standard deviation. The guiding hypothesis—proxy ranks and topological encodings are complementary—holds, though asymmetrically: removing the proxies costs three to five times what removing the topology does. The contribution is thus as much *selecting* a non-redundant proxy subset as fusing it with structure, and that selection transfers across datasets and spaces.

Limitations. Our contribution is narrower than "better NAS". ZAPS excels at recovering the top-100 *set* under a constrained budget, but does not separate from BANANAS on returning the single best CIFAR-10 architecture, where both reach the optimum in about $78\,\%$ of seeds at $B = 200$ and all by $B = 400$ ($p = 0.25$ on regret). The advantage also has a budget window: on CIFAR-100 the P@100 gap falls from $+15.4$ to $+0.6$ pp between $B = 200$ and $B = 500$—though not on NAS-Bench-101 (Section 4.5), verified on two spaces only. Finally, evaluation is confined to tabular benchmarks with savings unverified for real training, and the encoding assumes fixed cell topology.

Reproducibility statement. All experiments are table lookups on public benchmarks with no network training, so the reported numbers are exactly reproducible from the released artifacts. Complete hyperparameters are listed in Appendix C; code, run scripts, and per-seed results for all 200 seeds and all budgets will be released publicly upon acceptance.

Ethics statement. This work lowers the computational cost of architecture selection and may thereby broaden access to neural architecture design. We identify no direct negative societal impact: the study uses only public benchmark data, involves no human subjects or personally identifying information, and requires no GPU training.

AI use statement. A large language model (Claude, Anthropic) was used as an assistant throughout this work, and we describe its role by task rather than in general terms.

Research idea and method design. The four-stage pipeline, the anti-redundancy criterion of PROXY-FIT, the two-phase initialization and the choice to concatenate proxy ranks with topological encodings are the authors' own. The model did not propose the method.

Implementation. The authors wrote the original implementation of ZAPS and of the baselines. The model subsequently audited that code and identified the following defects, all in the baselines: BANANAS used a one-hot rather than a path encoding and optimized its acquisition over the full space rather than over mutations; REA counted duplicate evaluations against its budget on NAS-Bench-201 and, on NAS-Bench-101, used a projection that returned the parent in $99\,\%$ of mutations; TPE contained a fatal name error on NAS-Bench-201 and lost its exploitation component on NAS-Bench-101; one metric routine could loop indefinitely; and the Random Search generator had been lost. The model wrote the corrections and the replacement Random Search script, which the authors reviewed. All reported baseline numbers come from the corrected implementations.

Experiment design. The fusion ablation of Section 4.4—the proxies-only and topology-only variants that test this paper's central hypothesis—was proposed by the model after it observed that no existing experiment isolated the contribution of the two feature blocks. The authors approved and ran it. The model also proposed the significance testing and effect-size reporting.

Analysis and writing. The model produced the figure-generation scripts, ran the statistical tests, and drafted substantial portions of the text, which the authors edited. Because a language model can restate a number incorrectly, every numerical claim in this paper was subsequently re-derived from the result files by a verification script rather than accepted as written. That audit was run repeatedly as the results were finalized, and every discrepancy it surfaced—in reported means, effect sizes, significance thresholds and figure captions—was corrected against the result files. The verification scripts are released with the code.

What the model did not do. It did not generate data, run experiments without authorization, or produce any result reported here that is not backed by a committed CSV file. The authors take full responsibility for the contents of this submission.

REFERENCES

Mohamed S Abdelfattah, Abhinav Mehrotra, Yusuf Ozkan, and Nicholas D Lane. Zero-cost proxies for lightweight nas. In *International Conference on Learning Representations*, 2021.

Yash Akhauri, Juan Pablo Munoz, Nilesh Jain, and Ravishankar Iyer. EZNAS: Evolving zero-cost proxies for neural architecture scoring. In *Advances in Neural Information Processing Systems*, 2022.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, volume 24, 2011.

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

Peijie Dong, Lujun Li, Xiang Liu, Zhenheng Tang, Xuebo Liu, Qiang Wang, and Xiaowen Chu. LPZero: Language model zero-cost proxy search from zero. *arXiv preprint arXiv:2410.04808*, 2024.

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

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

Yi-Cheng Huang, Wei-Hua Li, Chih-Han Tsou, Jun-Cheng Chen, and Chu-Song Chen. UP-NAS: Unified proxy for neural architecture search. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR-NAS)*, 2024.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*, 2019.

Guihong Li, Yuedong Yang, Kartikeya Bhardwaj, and Radu Marculescu. Zico: Zero-shot nas via inverse coefficient of variation on gradients. In *International Conference on Learning Representations*, 2023.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

Lizheng Ma, Jiaxu Cui, and Bo Yang. Deep neural architecture search with deep graph Bayesian optimization. *arXiv preprint arXiv:1905.06159*, 2019.

Joe Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. In *International Conference on Machine Learning*, pp. 7588–7598. PMLR, 2021.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI Conference on Artificial Intelligence*, pp. 4780–4789, 2019.

Binxin Ru, Xingchen Wan, Xiaowen Dong, and Michael Osborne. Interpretable neural architecture search via Bayesian optimisation with Weisfeiler-Lehman kernels. In *International Conference on Learning Representations*, 2021.

Hidenori Tanaka, Daniel Kunin, Daniel LK Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6377–6389, 2020.

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

Chris Ying, Aaron Klein, Esteban Real, Eric Christiansen, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning*, pp. 7105–7114. PMLR, 2019.

Arber Zela, Julien Siems, Lucas Zimmer, Jovita Lukasik, Margret Keuper, and Frank Hutter. NAS-Bench-Suite-Zero: Accelerating research on zero cost proxies. In *Advances in Neural Information Processing Systems*, 2022.

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

A Baseline Implementation Details

All baselines are our own implementations. Table 4 records, for each, the settings we adopted and the behavioral check we used to verify it. The global optima used to compute regret are $91.57\,\%$ (CIFAR-10), $73.26\,\%$ (CIFAR-100) and $47.33\,\%$ (ImageNet16-120) on NAS-Bench-201, and $94.72\,\%$ on NAS-Bench-101. Two conventions apply throughout and are worth stating explicitly, since they affect comparability.

Budget counts distinct architectures. Re-evaluating an architecture already seen costs nothing, since its accuracy is known. This rule favours REA, whose mutations increasingly return architectures it has already evaluated: charging every proposal would cost it $19.3\,\%$ of its budget at $B = 200$ and $41.6\,\%$ at $B = 500$.

Projection onto the search space. NAS-Bench-101 encodes a cell in 56 bits, i.e. 2^{56} possible vectors; its validity constraints (connectivity, at most nine edges, a canonical form) leave $423{,}624$ distinct architectures, and NAS-Bench-Suite-Zero provides proxy values for a subset of $9{,}781$ of them, on which our entire evaluation is run. A mutated encoding therefore almost never falls in this pool and must be projected back onto it. The mutant lies at Hamming distance one from its parent, which is its nearest neighbour in the pool in $97\,\%$ of cases, so the projection excludes it: REA then draws the child uniformly among the eight nearest architectures, and TPE takes the nearest one not yet evaluated. Without that exclusion a nearest-neighbour projection returns the parent almost every time, so that $98\,\%$ of REA's mutations fall back on a random draw and its P@100 drops to that of Random Search. Even with it, about a third of REA's mutations reach an architecture already evaluated and are replaced by a random draw.

Table 4: Baseline settings and the behavioral check used to verify each implementation.

Method	**Settings**	**Verification**
Random Search	sampling without replacement	matches $B \times 100/N$ within 0.2 pp
Local Search	one-operation neighbourhood, steepest ascent, random restart on no improvement	$91.34\,\%$ at $B = 200$, slightly below the $\approx 91.5\,\%$ usually reported
REA	population 50, tournament 10, FIFO removal, one-operation mutation	$91.47\,\%$ at $B = 200$, matching published values
BANANAS	path encoding, $5\times$ ten-layer MLP, independent Thompson sampling, mutation-based candidates	attains the global optimum by $B = 500$
TPE	$\gamma = 0.25$, 64 candidates, discrete KDE per position	P@100 monotone in B on both datasets

B Algorithms

Algorithm 1 ProxyFit: greedy anti-redundant proxy selection

Require: Proxy pool P, redundancy threshold $\tau = 0.85$, target size $K = 6$
Ensure: Selected subset $\Pi \subseteq P$

```
Sort P by |ρ(p, f)| in decreasing order
Π ← ∅
for each p in sorted order do
    if max_{q∈Π} |ρ(p, q)| < τ then
        Π ← Π ∪ {p}
    end if
    if |Π| = K then break
    end if
end for
return Π
```

Algorithm 2 Bootstrapped online re-selection

Require: Labeled set $\mathcal{L}_t$, proxy pool P, $B_{\text{boot}} = 3$, $k_{\min} = 4$, $k_{\max} = 10$, threshold $\tau = 0.85$
Ensure: Active subset $\Pi_t \subseteq P$

$k \leftarrow \max\big(k_{\min}, \min(k_{\max}, \lfloor |\mathcal{L}_t|/15 \rfloor + 4)\big)$
votes$[p] \leftarrow 0$ for all $p \in P$
for $b = 1$ **to** B_{boot} **do**
    $\mathcal{L}^b \leftarrow$ bootstrap resample of $\mathcal{L}_t$; estimate all ρ on $\mathcal{L}^b$
    Sort P by $|\rho(p, f)|$ decreasing; $\Pi^b \leftarrow \{\text{first proxy}\}$
    for each remaining p in sorted order **do**
        stop if $|\Pi^b| = k_{\max}$, or if $J(p \mid \Pi^b) < 0$ and $|\Pi^b| \geq k_{\min}$
        $\Pi^b \leftarrow \Pi^b \cup \{p\}$ **if** $\max_{q \in \Pi^b} |\rho(p, q)| < \tau$
    end for
    votes$[p]$ += 1 for each $p \in \Pi^b$
end for
return the k proxies with the highest vote counts

Algorithm 3 ZAPS: full active-learning loop

Require: Search space $\mathcal{A}$, budget B, iterations $T = 2$, exploration weight β
Ensure: Predicted top-100 architectures

$\Pi_{\text{init}} \leftarrow \textsc{ProxyFit}(P)$ ▷ offline, once
$n_0 \leftarrow \max(10, \lfloor B/4 \rfloor)$; $b \leftarrow \max\big(1, \lfloor (B - n_0)/T \rfloor\big)$
$\mathcal{L}_0 \leftarrow$ hybrid K-means initialization (n_0 architectures) using Π_{init}; evaluate f on $\mathcal{L}_0$
for $t = 0$ **to** $T - 1$ **do**
    $\Pi_t \leftarrow$ bootstrapped re-selection on $\mathcal{L}_t$ (Algorithm 2)
    Build $\phi(a) = [\hat{\mathbf{r}}(a); \mathbf{e}(a)]$ for all $a \in \mathcal{A}$
    Fit the XGBoost ensemble on $\{(\phi(a), f(a))\}_{a \in \mathcal{L}_t}$ to obtain $\mu(\cdot), \sigma(\cdot)$
    $\mathcal{B}_t \leftarrow$ the b unevaluated architectures maximizing $\mu(a) + \beta\sigma(a)$
    Evaluate $f(a)$ for $a \in \mathcal{B}_t$; $\mathcal{L}_{t+1} \leftarrow \mathcal{L}_t \cup \mathcal{B}_t$
end for
$\Pi_T \leftarrow$ re-selection on $\mathcal{L}_T$; refit the ensemble on $\mathcal{L}_T$ to obtain $\mu_{\text{final}}(\cdot)$
return the 100 architectures of $\mathcal{A}$ with the largest $\mu_{\text{final}}(a)$

The loop consumes $n_0 + Tb \leq B$ true evaluations, so the budget is never exceeded; the floor in b leaves at most one evaluation unspent. The ranking returned is produced by a final refit on the complete labeled set $\mathcal{L}_T$, after the last batch has been evaluated.

C HYPERPARAMETERS

Table 5 lists every hyperparameter of ZAPS. All values are held fixed across datasets, budgets, and search spaces; none is tuned per experiment. Every reported figure averages the seeds $0, \ldots, 199$, used identically by ZAPS and by every search baseline; the proxy-combination schemes of Appendix E are deterministic and are run once.

Table 5: Complete hyperparameter settings for ZAPS.

Stage	Hyperparameter	Value
ProxyFit	Redundancy threshold τ	0.85
	Target subset size K	6
Initialization	Initialization budget n_0	$\max(10, \lfloor B/4 \rfloor)$
	Exploitation fraction	top 30 % by proxy rank
	Clusters per phase	$n_0/2$
	K-means restarts / max iterations	3 / 100
Re-selection	Bootstrap resamples B_{boot}	3
	Minimum subset size $k_{\min}$	4
	Maximum subset size $k_{\max}$	10
	Target size k	$\max(k_{\min}, \min(k_{\max}, \lfloor \lvert\mathcal{L}_t\rvert/15 \rfloor + 4))$
	Redundancy threshold τ	0.85
Surrogate	Ensemble members B_{ens}	3
	Tree method / objective	histogram / squared error
	Trees per regressor	50
	Maximum tree depth	5
	Learning rate	0.05
	Row subsample	0.8
	Column subsample per tree	0.8
	Minimum child weight	3
	L_1 regularization α	0.1
Active loop	Exploration weight β	0.5
	Iterations T	2
	Batch size b	$\lfloor (B - n_0)/T \rfloor$

D Ablation Curves

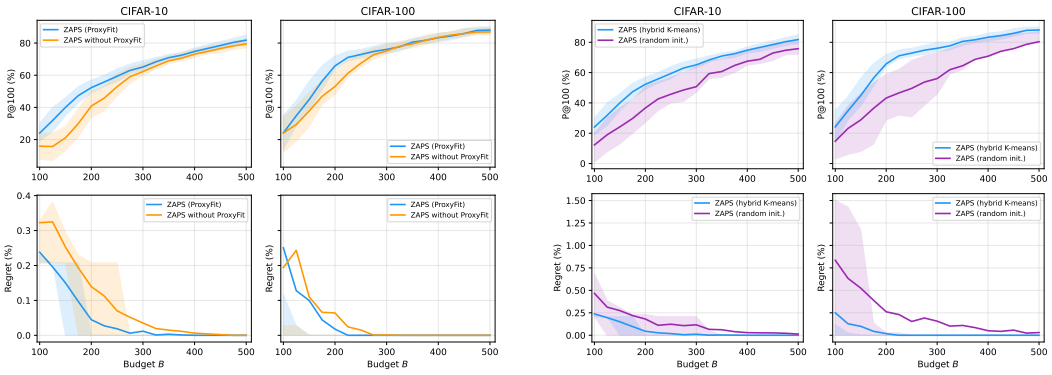


Figure 3: Ablations across the full budget range on CIFAR-10 and CIFAR-100. Lines are means over 200 seeds and shaded bands interquartile ranges. Left: the ProxyFit subset of six proxies against the full pool of thirteen. Right: hybrid K-means initialization against uniform random sampling. Table 2 reports the values at $B = 200$.

E ZERO-COST PROXY COMBINATION

Table 6: Zero-cost proxy schemes on NAS-Bench-201, each spending $B = 200$ true evaluations: the scheme ranks the space from proxy scores, then trains its top 200; P@100 counts the true top-100 among the architectures trained. *Best single* is chosen by highest Spearman ρ against accuracy, using oracle knowledge unavailable to a practitioner. The last row aggregates the three proxies recommended by Abdelfattah et al. (2021) by summing their normalized ranks (Borda count); we use this rather than their pairwise majority vote, which does not in general induce a total order over the space. For reference, ZAPS reaches 52.3 and 65.8 % at the same budget.

	CIFAR-10		**CIFAR-100**	
Method ($B = 200$)	P@100 (%)	Regret (%)	P@100 (%)	Regret (%)
`nwot` (best single, oracle)	0.0	0.91	7.0	1.50
`synflow` (fixed)	22.0	**0.00**	36.0	0.03
Rank product, all 13	0.0	1.34	6.0	1.50
Rank product, positive only	0.0	0.91	8.0	1.50
Borda, {`synflow`,`jacov`,`snip`}	0.0	1.06	5.0	1.17

F FUSION ABLATION CURVES

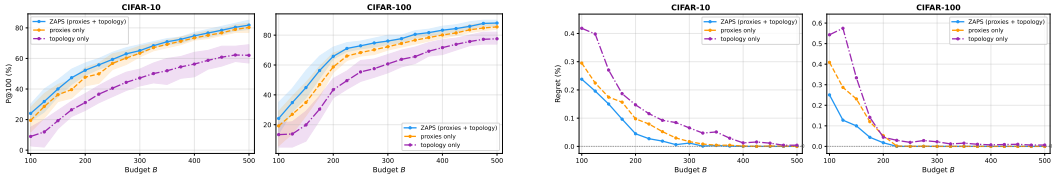


Figure 4: Fusion ablation across the full budget range, 200 seeds. Only the surrogate's input differs between the three variants. The topology-only variant never closes its gap; the proxies-only variant converges towards the full pipeline as labels accumulate. Regret is shown on a linear scale because ZAPS reaches exactly zero at several budgets, which a logarithmic axis cannot display.

G PROXY POOL AND CORRELATION STRUCTURE

Table 7: The 13 zero-cost proxies of NAS-Bench-Suite-Zero, grouped by family. The last column gives the range of Spearman ρ with true accuracy on NAS-Bench-201 / CIFAR-10.

Family	**Proxies**	**Spearman ρ**
Deterministic	`params`, `flops`, `l2_norm`, `synflow`	0.68–0.73
Activation	`nwot`, `jacov`, `epe_nas`, `zen`, `plain`	−0.27–0.77
Gradient	`fisher`, `snip`, `grad_norm`, `grasp`	0.51–0.60

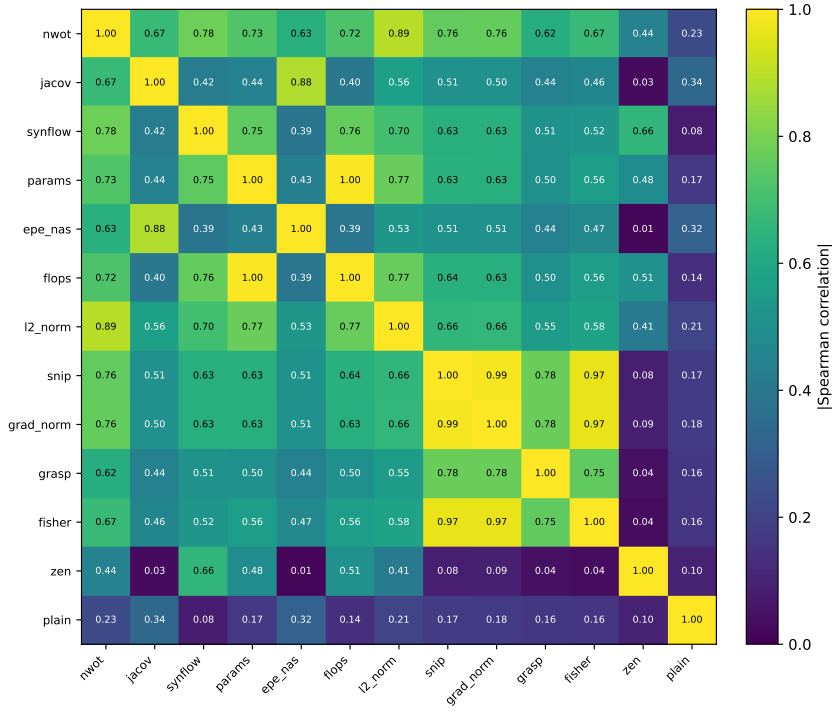


Figure 5: Absolute Spearman correlation between proxy pairs on NAS-Bench-201 / CIFAR-10, with proxies ordered by decreasing relevance—the order in which PROXYFIT examines them. Redundancy is assessed on $|\rho|$, since two strongly anti-correlated proxies are as redundant as two strongly correlated ones. Highly correlated pairs such as `flops–params` ($|\rho| = 0.995$) are the redundancies PROXYFIT is designed to eliminate.

H CONVERGENCE AND RANKING QUALITY

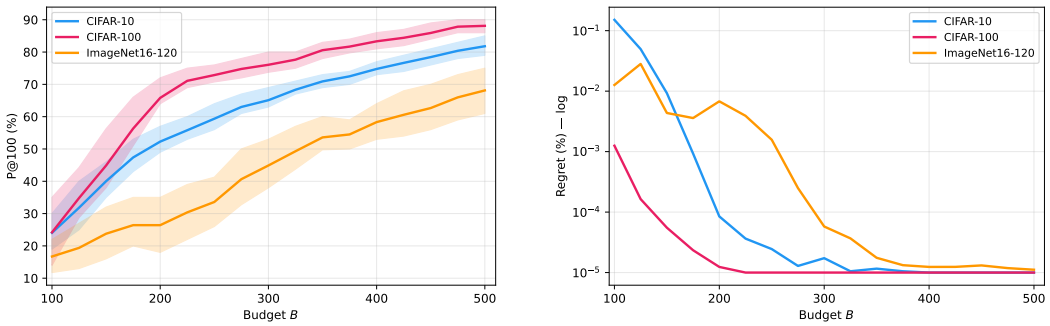


Figure 6: Convergence of ZAPS in isolation: P@100 (left) and regret (right) against budget over 200 seeds; the three curves are CIFAR-10, CIFAR-100 and ImageNet16-120. P@100 lines are means with interquartile bands; regret lines are geometric means floored at 10^{-5}, so a curve resting on the floor marks budgets at which every seed reaches the global optimum.

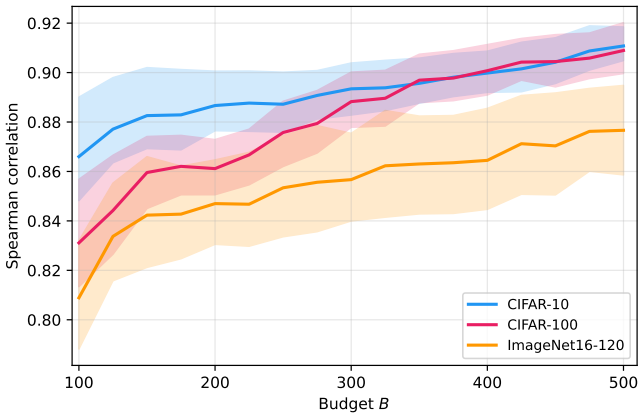


Figure 7: Spearman correlation between the surrogate ranking and the ground-truth ranking over all 15,625 architectures of NAS-Bench-201. Lines are means over 200 seeds and shaded bands interquartile ranges; the three curves are CIFAR-10, CIFAR-100 and ImageNet16-120. High global correlation indicates that the surrogate is not merely accurate near the optimum.

I Head-to-Head Comparison with the Strongest Baselines

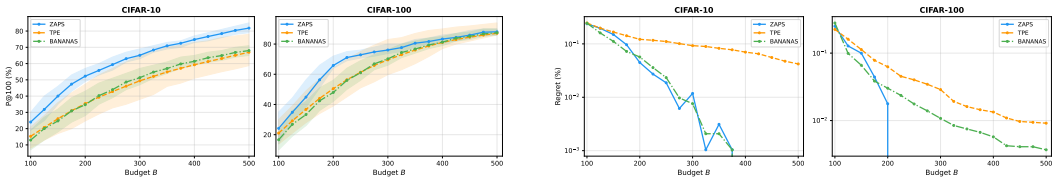


Figure 8: ZAPS against TPE and BANANAS, the two strongest baselines, across the full budget range. ZAPS leads at every budget on both datasets; the gap narrows on CIFAR-100 as the budget grows, from $+15.4$ pp over TPE at $B = 200$ to $+0.6$ pp over BANANAS at $B = 500$. Its interquartile band is the narrowest at nearly every budget: at $B = 500$ on CIFAR-100 the per-seed standard deviation is 3.7 pp for ZAPS against 11.1 for TPE and 5.2 for BANANAS. Regret uses a logarithmic axis, on which budgets where the mean regret is exactly zero cannot be drawn and are omitted.

J Cross-Dataset Transfer

Table 8: P@100 (%) when ProxyFit is calibrated on the target dataset (*native*) versus transferred unchanged from CIFAR-10 (*transferred*). Mean over 200 seeds. A transferred subset uses no ground-truth accuracy from the target dataset.

	CIFAR-100		**ImageNet16-120**	
B	native	transferred	native	transferred
100	24.2	**32.7**	**16.7**	14.8
200	**65.8**	62.0	26.4	**27.5**
300	76.1	**76.5**	44.9	**48.9**
500	**88.1**	87.1	68.1	**70.7**

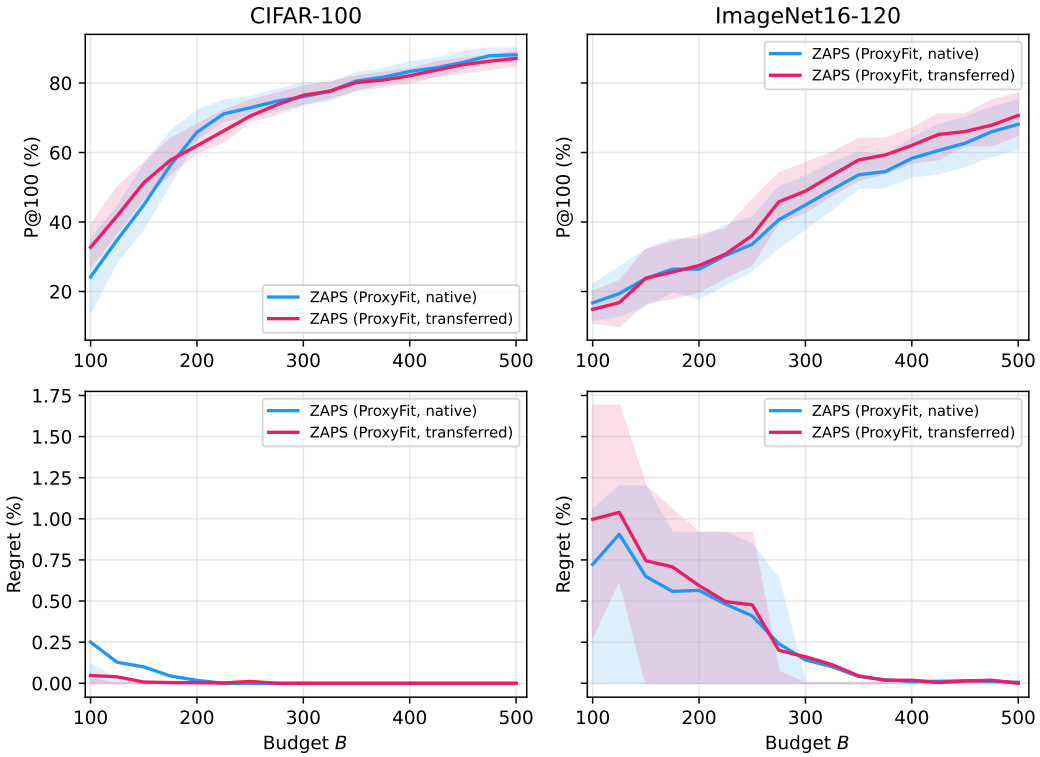


Figure 9: Cross-dataset transfer across the full budget range. The proxy subset calibrated on CIFAR-10 is applied unchanged to CIFAR-100 and ImageNet16-120 (*transferred*) and compared against a subset calibrated on the target dataset itself (*native*). Table 8 reports the values at four budgets.

K Results on NAS-Bench-101

Table 9: P@100 (%) on NAS-Bench-101 / CIFAR-10, 200 seeds, unified metric.

Method	$B = 100$	$B = 200$	$B = 300$	$B = 500$
Random Search	1.0	2.0	3.0	5.0
REA	2.5	7.2	12.0	20.1
TPE	8.8	18.4	26.3	37.0
ZAPS	**13.1**	**31.5**	**46.7**	**63.0**

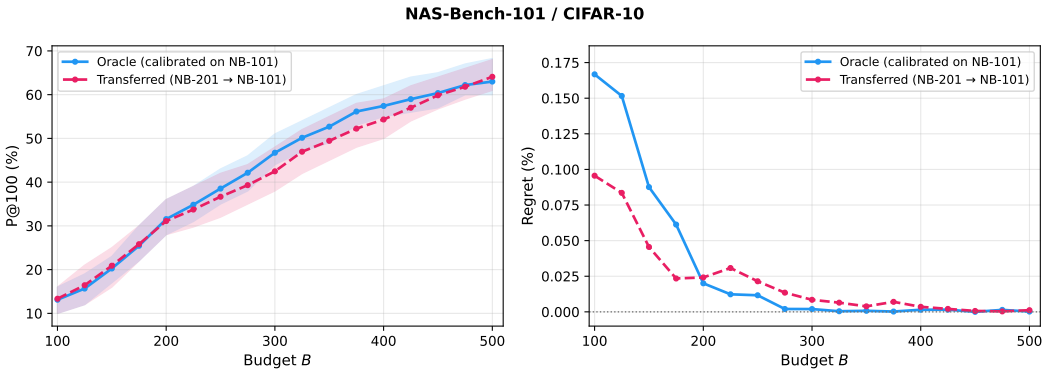


Figure 10: Cross-space transfer. The proxy subset calibrated on NAS-Bench-201 is applied unchanged to NAS-Bench-101 (*transferred*) and compared against a subset calibrated directly on NAS-Bench-101 (*oracle*). Across the whole budget range the two differ by at most 4.2 pp in P@100 and 0.07 pp in regret, so the calibration transfers without meaningful loss.

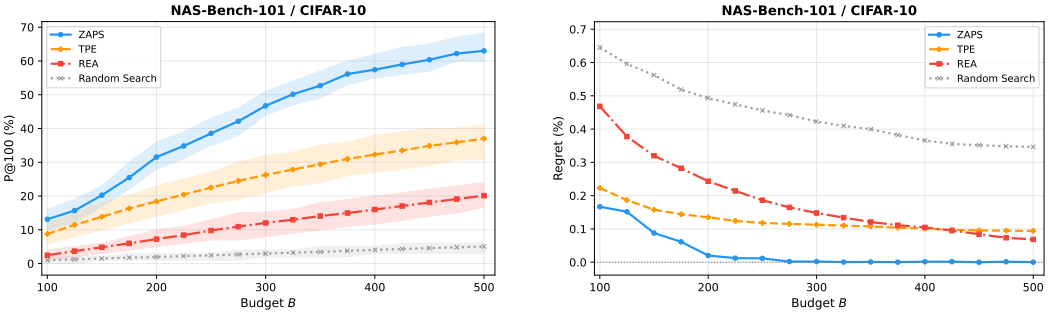


Figure 11: P@100 (left) and regret (right) on NAS-Bench-101 / CIFAR-10 over 200 seeds, comparing ZAPS to the baselines of Table 9. Lines are means, with interquartile bands for P@100. Unlike NAS-Bench-201, the gap widens with the budget and no method approaches saturation.