NICE-CV: NEURAL INFERENCE- AND COST-EFFICIENT ARCHITECTURE SEARCH FOR COMPUTER VISION MODELS WITH ONLINE PROXY CALIBRATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Efficient neural architecture search (NAS) has two coupled objectives: the discovered network must maintain strong performance while achieving low inference latency, and the search must cost training time selectively. Zero-cost proxies reduce search cost, but their reliability varies across tasks and training fidelities, while fixed proxy combinations cannot adapt to new validation evidence. Therefore, we introduce NICE-CV, a multi-fidelity NAS method for inference- and search-efficient architecture discovery. To the best of our knowledge, NICE-CV is the first multi-fidelity NAS method to treat zero-cost proxy reliability as an online search state, continuously updating it from purchased validation evidence to guide architecture–fidelity decisions that jointly account for measured inference latency and wall-clock search cost. Concretely, NICE-CV converts heterogeneous proxy scores into comparable ranks, updates proxy reliability from pairwise validation comparisons, and combines the calibrated proxy evidence with cross-fidelity observations to guide subsequent search decisions. Each legal start or promotion is scored by posterior expected hypervolume improvement per predicted search second, while measured inference latency is incorporated directly into the search objective. The resulting search is invariant to monotone transformations of proxy scores, and its online reliability learner enjoys a no-regret guarantee relative to the best fixed proxy in hindsight. Across five complete tabular NAS benchmark families, NICE-CV achieves the highest mean normalized hypervolume ratio (HVR) among executable search methods once online feedback becomes available. On recorded NAS-Bench-201 fidelities, it achieves the highest HVR among the evaluated multi-fidelity methods at 0.9360 while using substantially less search time than full-fidelity baselines. Training-based searches across classification, segmentation, and detection further show that NICE-CV achieves the highest mean HVR on all four tasks; compared with the fastest baseline on each task, it requires only marginal additional search time while consistently recovering higher-quality architecture sets. These results show that continuously adapting inexpensive proxy guidance to task-specific feedback can improve both inference-efficient architecture discovery and wall-clock-efficient search, yielding stronger quality–latency tradeoffs under limited search budgets.

1 INTRODUCTION

Neural architecture search (NAS) is often formulated around maximizing validation accuracy, but practical vision deployment imposes two coupled efficiency requirements. First, the discovered architecture should maintain strong task performance while keeping inference latency low. Second, the search procedure itself should use limited training time efficiently. These objectives are related: wider and deeper networks may improve classification, segmentation, or detection quality, but they also increase inference latency, memory traffic, and the cost of every subsequent training evaluation. As a result, accuracy-first search followed by latency filtering can spend substantial search budget on architectures that are ultimately unattractive for deployment.

Zero-cost (ZC) proxies offer an inexpensive way to rank architectures without full training (Abdelfattah et al., 2021; Mellor et al., 2021; Tanaka et al., 2020), but their reliability is not stable across

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.

tasks, architecture spaces, or training fidelities. A proxy that is informative for one task or stage of training may be misleading for another, while fixed proxy combinations cannot adapt when new task-specific evidence becomes available. Short training provides more direct validation evidence, but introduces a second problem: under a limited budget, the search must decide not only which architectures to evaluate, but also whether to start a new candidate, promote an existing one to a higher fidelity, or stop allocating resources to it. Accuracy-based Successive Halving (SH) (Li et al., 2018) addresses training allocation, but it does not revise the reliability of its initial guidance and can discard low-latency candidates that remain attractive under a quality–latency tradeoff. This raises a practical question: *how should a NAS procedure continuously update which cheap signals it trusts while deciding where to spend the next unit of training time, so that a limited search budget yields architectures with strong task quality and low inference latency?*

To address this question, we introduce NICE-CV, **N**eural **I**nference- and **C**ost-**E**fficient architecture search for computer vision models. To the best of our knowledge, NICE-CV is the first multi-fidelity NAS method to treat zero-cost proxy reliability as an online search state, continuously updating it from purchased validation evidence to guide architecture–fidelity decisions that jointly account for measured inference latency and wall-clock search cost. Concretely, NICE-CV converts heterogeneous proxy scores into empirical ranks, measures each proxy's pairwise agreement with observed validation outcomes, and updates a reliability distribution by exponential weights. These calibrated proxy signals are combined with fidelity-dependent validation evidence to estimate architecture quality across fidelities. At each step, NICE-CV scores every legal start or promotion by expected hypervolume improvement (EHVI) per predicted incremental second and executes the highest-value affordable action. Because measured inference latency is included directly in the objective rather than applied as a post-search filter, low-latency architectures remain valuable even when their predicted accuracy is not maximal. As validation evidence accumulates, misleading proxies lose influence and subsequent training budget is redirected toward more promising architecture–fidelity decisions.

We evaluate NICE-CV through three complementary levels of evidence: complete tabular NAS benchmarks, recorded multi-fidelity search, and training-based search on real computer-vision tasks. Search quality is measured using normalized hypervolume ratio (HVR), which summarizes coverage of the quality–latency tradeoff relative to a reference frontier. Across five tabular NAS benchmark families covering 77 task–cost conditions and 100 paired seeds, NICE-CV achieves the highest mean HVR at every feedback-enabled budget. On recorded NAS-Bench-201 fidelities, NICE-CV achieves the highest HVR among evaluated multi-fidelity methods at 0.9360 while also obtaining the lowest recorded search time. Training-based searches span CIFAR-100 and Tiny-ImageNet classification, Oxford-IIIT Pet segmentation, and Penn-Fudan detection, with deployment latency measured on an RTX A6000. Compared with both classical search baselines and recent NAS methods, NICE-CV achieves the highest mean HVR on all four tasks; faster baselines generally recover lower-quality architecture sets, while NICE-CV obtains substantially stronger search quality with only modest additional search time. On Tiny-ImageNet, for example, it improves HVR over Early-SH from 0.8355 to 0.8808 while slightly reducing search time from 3.871 to 3.858 minutes. Together, these results show that continuously adapting proxy guidance to task-specific validation feedback can improve both inference-efficient architecture discovery and wall-clock-efficient search under limited budgets.

Our contributions are:

- We introduce an online rank-calibration mechanism that first treats zero-cost proxy reliability as a dynamic search state and updates it from pairwise validation evidence, allowing proxy guidance to adapt across tasks and fidelities.
- We develop a unified multi-fidelity architecture–fidelity acquisition rule that couples calibrated proxy evidence, measured inference latency, and predicted incremental training cost to determine which candidate to start or promote next.
- We provide monotone-invariance and no-regret analyses for the reliability learner, together with controlled ablations showing that online reliability adaptation accounts for most of the observed HVR improvement.
- We evaluate NICE-CV across five complete tabular NAS benchmark families and four training-based computer-vision tasks, comparing classical search strategies and recent NAS methods, including AZ-NAS, MF-NAS, and R-MF-NAS, in terms of both quality–latency search quality and wall-clock search cost.

2 Related Work

Efficient and hardware-aware NAS. DARTS and SNAS relax discrete architecture choices (Liu et al., 2019; Xie et al., 2019), while ProxylessNAS, FBNet, and Once-for-All incorporate device costs or amortize training through weight sharing (Cai et al., 2019; Wu et al., 2019; Cai et al., 2020).

Zero-cost estimation. SynFlow measures initialization-time gradient flow (Tanaka et al., 2020), NASWOT activation-pattern diversity (Mellor et al., 2021), and GradNorm/SNIP/Fisher parameter sensitivity (Abdelfattah et al., 2021). Their rankings vary strongly across search spaces and tasks (Krishnakumar et al., 2022). AZ-NAS reduces single-proxy variance through expressivity, progressivity, trainability, and complexity ranks (Lee & Ham, 2024); UP-NAS learns a multi-proxy predictor before search (Huang et al., 2024). Both demonstrate the value of complementary views.

Multi-fidelity and multi-objective search. SH/Hyperband allocate larger budgets to smaller cohorts, and BOHB learns density-ratio proposals (Li et al., 2018; Falkner et al., 2018). RANK-NOSH learns rankings from non-uniform partial training (Wang et al., 2021). MF-NAS uses a selected ZC proxy for local search before SH; R-MF-NAS diversifies the SH pool with start and intermediate trajectory states (Phan & Luong, 2026). Multi-objective evolutionary methods instead retain non-dominated candidates (Deb et al., 2002); hypervolume rewards both convergence and frontier coverage (Zitzler & Thiele, 1999). NICE-CV uses hypervolume twice: as the final quality indicator and as the value of a fidelity observation.

Benchmarks. NAS-Bench-101/201 and NATS-Bench define finite architecture universes (Ying et al., 2019; Dong & Yang, 2020; Dong et al., 2022); TransNAS-Bench broadens task type, and HW-NAS-Bench supplies device measurements (Duan et al., 2021; Li et al., 2021). We exploit their exact reference hypervolumes and paired seeds, then test actual MobileNet-like training following NAS reporting guidance (Lindauer & Hutter, 2020; Li & Talwalkar, 2020). NICE-CV replaces fixed scoring with online reliability, explicit promotion actions, measured CV latency, and search guided by quality–latency frontier improvement spanning classification, segmentation, and detection.

3 Why Multi-Fidelity NAS Needs Online Proxy Calibration

A limited-budget NAS procedure must decide which architectures to evaluate, how far to train them, and which inexpensive signals to trust before sufficient task-specific evidence is available. Existing zero-cost and multi-fidelity methods usually treat these components separately: proxies provide fixed initial guidance, while successive-halving-style methods allocate training according to observed validation performance. This ignores an important feedback loop, because every purchased validation observation can also reveal whether the proxies guiding the search remain reliable.

Proxy reliability varies across tasks and fidelities. Let $z_j(a)$ denote proxy j's score for architecture a, and let $y_{d,f}(a)$ denote validation quality on task d at fidelity f. We define proxy reliability as

$$R_{j,d,f} = 2 \Pr_{a,b\sim\mathcal{P}}[(z_j(a) - z_j(b))(y_{d,f}(a) - y_{d,f}(b)) > 0] - 1. \tag{1}$$

Without ties, $R = 1$ denotes perfect ordering agreement and $R = 0$ chance agreement. Because $R_{j,d,f}$ changes with both task and fidelity, a fixed proxy combination $S(a) = \sum_{j=1}^{J} \bar{w}_j r_j(a)$ can incur excess ranking loss relative to condition-specific weights:

$$\mathcal{E}_{\text{static}} = \min_{\bar{\mathbf{w}}\in\Delta^{J-1}} \mathbb{E}_{d,f}\left[\mathcal{L}_{d,f}(\bar{\mathbf{w}}) - \min_{\mathbf{w}\in\Delta^{J-1}} \mathcal{L}_{d,f}(\mathbf{w})\right]. \tag{2}$$

Validation evidence obtained during search can therefore be used to continuously revise which proxies should be trusted.

Static filtering can make early mistakes irreversible. Let $G_z(k)$ contain the k architectures retained by a static score z, and let $\mathcal{P}^\star$ be the terminal quality–cost efficient set. Then

$$\widehat{\mathcal{P}}_T \subseteq G_z(k) \quad\Longrightarrow\quad \mathrm{HV}(\widehat{\mathcal{P}}_T) \le \mathrm{HV}(G_z(k)). \tag{3}$$

Once an architecture is excluded by static proxy ranking, later successive halving cannot recover it. This is especially undesirable when a candidate offers lower inference latency despite slightly lower predicted accuracy. Multi-fidelity observations should therefore not only determine promotion, but also update the proxy guidance used for subsequent decisions.

Search decisions must account for both inference and search cost. For action $x_t = (a_t, f_t)$ with incremental search-time cost $\tau(x_t)$, a limited-budget policy solves

$$\max_{\pi} \mathbb{E}_{\pi}\Big[\mathrm{HV}(\widehat{\mathcal{P}}_T)\Big] \quad \text{s.t.} \quad \sum_{t=1}^{T} \tau(x_t) \le B. \tag{4}$$

Here, inference latency characterizes the deployment efficiency of an architecture, whereas $\tau(x_t)$ measures the wall-clock cost of obtaining additional search evidence. These two costs therefore play different roles: a low-latency architecture may remain valuable even if its predicted quality is not maximal, while an expensive high-fidelity evaluation may be unattractive if it provides only limited expected improvement. The value of an architecture–fidelity action should therefore depend jointly on expected quality–latency improvement and the incremental search time required to obtain it.

These observations motivate NICE-CV. Rather than treating zero-cost proxies as fixed heuristics used only before training, NICE-CV maintains proxy reliability as an online search state and updates it whenever new validation evidence is purchased. The updated reliability is then combined with fidelity-dependent observations to guide subsequent start and promotion decisions. Measured inference latency enters the architecture objective directly, while predicted incremental training time determines the cost of each search action. In this way, NICE-CV couples which proxies to trust, which architectures to evaluate, and how much training to allocate within a single limited-budget multi-fidelity search process.

These observations motivate NICE-CV: proxy reliability is maintained as an online search state and updated whenever new validation evidence is purchased. The updated reliability then guides subsequent start and promotion decisions, while measured inference latency and predicted incremental training time jointly determine how limited search budget is allocated.

4 METHOD

4.1 COST-AWARE MULTI-OBJECTIVE SEARCH

Let $\mathcal{P} = \{a_i\}_{i=1}^{N} \subset \mathcal{A}$ be a pool of N candidates from architecture space $\mathcal{A}$, $c(a) > 0$ the measured deployment cost of architecture a, and $\mathcal{F} = \{f_1 < \cdots < f_L = F\}$ a ladder of L training fidelities with terminal fidelity F. Training yields validation quality $y_f(a)$, with terminal quality $Q(a) = y_F(a)$; J initialization experts provide proxy scores $z_j(a)$. At step t, state $\mathcal{S}_t = (\mathcal{D}_t, \mathcal{B}_t, \mathbf{w}_t, \widehat{\tau}_t)$ stores observations, remaining search-time budget, expert reliability weights, and incremental-time predictions. Legal actions start an unseen candidate or promote an observed one:

$$\begin{aligned}\mathcal{X}_t = \{&(a, f_1) : a \in \mathcal{P} \text{ untrained}\} \\ &\cup \{(a, f_{\ell+1}) : (a, f_\ell, \cdot, \cdot) \in \mathcal{D}_t, (a, f_{\ell+1}, \cdot, \cdot) \notin \mathcal{D}_t,\ 1 \le \ell < L\}.\end{aligned} \tag{5}$$

Here, $\mathcal{X}_t$ is the legal action set, ℓ indexes fidelity levels, and dots denote observation fields not used in the membership test. The target is the terminal non-dominated set $\mathcal{P}^\star$: architectures $a \in \mathcal{P}$ for which no $b \in \mathcal{P}$ satisfies $Q(b) \ge Q(a)$ and $c(b) \le c(a)$ with at least one strict inequality.

4.2 RANKS AND ONLINE RELIABILITY

After orienting quality proxies so larger is better and fixing a tie order $\prec$, we use

$$r_j(a) = \frac{1}{N-1} \sum_{b \in \mathcal{P} \setminus \{a\}} \big[\mathbb{I}\{z_j(b) < z_j(a)\} + \mathbb{I}\{z_j(b) = z_j(a), b \prec a\}\big], \tag{6}$$

$$e(a) = 1 - \frac{1}{N-1} \sum_{b \in \mathcal{P} \setminus \{a\}} \big[\mathbb{I}\{c(b) < c(a)\} + \mathbb{I}\{c(b) = c(a), b \prec a\}\big]. \tag{7}$$

Here, $\mathbb{I}\{\cdot\}$ is the indicator function, $r_j(a)$ is the proxy rank, and $e(a)$ is the deployment-efficiency rank; both lie in $[0, 1]$, with larger values preferred.

Reliability is learned from order rather than potentially biased low-fidelity values. Let $\rho(f) = \sqrt{f/F}$ weight fidelity, and let $\chi(u, v)$ equal 0, $1/2$, or 1 for agreeing, tied, or reversed signs of its arguments. Observation (a_t, f_t, y_t), where $y_t = y_{f_t}(a_t)$, gives

$$\ell_{j,t} = \frac{1}{Z_t} \sum_{s<t} \rho(f_t)\rho(f_s)\chi(r_j(a_t) - r_j(a_s), y_t - y_s)\,, \tag{8}$$

$$w_{j,t+1} = \frac{w_{j,t}e^{-\eta\ell_{j,t}}}{\sum_{h=1}^{J} w_{h,t}e^{-\eta\ell_{h,t}}}, \qquad w_{j,1} = J^{-1}. \tag{9}$$

Here, s indexes earlier observations, $\ell_{j,t}$ is expert j's disagreement loss, $Z_t = \sum_{s<t} \rho(f_t)\rho(f_s)$ normalizes it, $\eta > 0$ is the learning rate, and h indexes experts. With no prior comparison, the weights remain unchanged. The tempered weights $\pi_{j,t} = w_{j,t}^{\gamma} / \sum_{h=1}^{J} w_{h,t}^{\gamma}$, with exponent $\gamma = 2$, emphasize more reliable experts.

Writing cumulative loss as $L_{j,t} = \sum_{s=1}^{t} \ell_{j,s}$ yields a generalized Bayesian posterior:

$$w_{j,t+1} = \frac{p_0(j)\exp(-\eta L_{j,t})}{\sum_{h=1}^{J} p_0(h)\exp(-\eta L_{h,t})}, \qquad \log\frac{w_{j,t+1}}{w_{h,t+1}} = \log\frac{p_0(j)}{p_0(h)} - \eta(L_{j,t} - L_{h,t}). \tag{10}$$

Here, $p_0(j)$ is the prior weight of expert j, equal to $1/J$ under uniform initialization. Each comparison therefore contributes additive log-evidence.

Let $r_t^{\mathrm{val}}(a)$ be the validation rank among comparable observations and $f(a)$ the current fidelity of a. Expert j predicts terminal rank

$$\mu_{j,t}(a) = \begin{cases} r_j(a), & a \text{ untrained}, \\ [1 - \rho(f(a))]r_j(a) + \rho(f(a))r_t^{\mathrm{val}}(a), & \text{otherwise}. \end{cases} \tag{11}$$

These predictions induce the distribution $p_t(q \mid a) = \sum_{j=1}^{J} \pi_{j,t}\delta_{\mu_{j,t}(a)}(q)$, where q denotes terminal rank and δ_m is a point mass at m. Their weighted variance, $U_t(a) = \sum_{j=1}^{J} \pi_{j,t}(\mu_{j,t}(a) - \sum_{h=1}^{J} \pi_{h,t}\mu_{h,t}(a))^2$, measures expert disagreement.

The interpolation has a minimum-variance interpretation. Suppose proxy rank r_j and validation rank r_f^{val} are conditionally unbiased estimates of terminal rank $q(a)$, with independent errors of variances $\sigma_{p,j}^2$ and σ_f^2. For the interpolated estimate $\widetilde{q} = (1 - \rho)r_j + \rho r_f^{\mathrm{val}}$, where $\rho \in [0, 1]$,

$$\mathrm{MSE}(\widetilde{q}) = (1 - \rho)^2\sigma_{p,j}^2 + \rho^2\sigma_f^2, \tag{12}$$

$$\rho_j^*(f) = \frac{\sigma_{p,j}^2}{\sigma_{p,j}^2 + \sigma_f^2}, \qquad \mathrm{MSE}(\widetilde{q}_{\rho^*}) = \frac{\sigma_{p,j}^2\sigma_f^2}{\sigma_{p,j}^2 + \sigma_f^2}. \tag{13}$$

Here, MSE is mean squared error relative to $q(a)$, and $\rho_j^*(f)$ is its minimizing interpolation weight. As validation variance decreases, the optimal weight shifts toward validation. Our $\rho(f) = \sqrt{f/F}$ approximates this behavior, while $\pi_{j,t}$ adapts proxy reliability online.

4.3 Expected hypervolume gain per second

Let $\mathcal{H}_t$ be the current non-dominated set in rank–efficiency coordinates. Define

$$I_t(q, a) = \mathrm{HV}(\mathcal{H}_t \cup \{(q, e(a))\}) - \mathrm{HV}(\mathcal{H}_t), \tag{14}$$

$$\mathrm{EHVI}_t(a) = \sum_{j=1}^{J} \pi_{j,t} I_t(\mu_{j,t}(a), a). \tag{15}$$

Here, HV is dominated hypervolume relative to the origin, $I_t(q, a)$ is the gain from adding candidate a at rank q, and $\mathrm{EHVI}_t(a)$ is its expectation under the discrete expert mixture.

For action $x = (a, f)$, NICE-CV selects

$$x_{t+1} \in \underset{x \in \mathcal{X}_t : \widehat{\tau}_t(x) \le \mathcal{B}_t}{\arg\max} \frac{\rho(f)\,\mathrm{EHVI}_t(a) + \lambda_t \sqrt{U_t(a)}}{\widehat{\tau}_t(x) + \varepsilon}, \qquad \lambda_t = \lambda_0 \sqrt{\mathcal{B}_t / \mathcal{B}_0}. \tag{16}$$

Here, $\widehat{\tau}_t(x)$ is predicted incremental search time, $\varepsilon > 0$ stabilizes division, $\mathcal{B}_0$ is the initial budget, and $\lambda_0 \ge 0$ sets the initial disagreement bonus. Its coefficient λ_t decreases with the remaining budget, encouraging exploration early. The fidelity weight and time denominator balance evaluation reliability against cost. After execution, the observations and acquisition are updated until no affordable action remains. Appendix B gives pseudocode and complexity.

4.4 Theoretical properties

Proposition 1 (Monotone invariance). *Strictly increasing transformations of any z_j or c preserve ranks, losses, reliability weights, and the complete action trajectory, subject to fixed ties and wall-clock estimates.*

Theorem 1 (No-regret reliability). *For $\ell_{j,t} \in [0, 1]$, uniform initialization, and Eq. equation 9,*

$$\sum_{t=1}^{T} \langle \mathbf{w}_t, \boldsymbol{\ell}_t \rangle - \min_j \sum_{t=1}^{T} \ell_{j,t} \le \frac{\log J}{\eta} + \frac{\eta T}{8}. \tag{17}$$

Here, T is the number of update rounds, $\boldsymbol{\ell}_t = (\ell_{1,t}, \ldots, \ell_{J,t})$, and the inner product is the reliability-weighted loss. At $\eta = \sqrt{8 \log J / T}$, average excess loss is $O(\sqrt{\log J / T})$; proofs are in Appendix A.

Corollary 1 (Exponential suppression). *If experts j and h satisfy $L_{j,t} - L_{h,t} \ge t\delta$, where $\delta > 0$ is a lower bound on their average loss gap, then*

$$\frac{w_{j,t+1}}{w_{h,t+1}} \le \frac{p_0(j)}{p_0(h)} e^{-\eta t \delta}. \tag{18}$$

Persistently misleading proxies therefore lose relative weight exponentially.

4.5 Evaluation metric

For evaluation, define $u(a) = (Q(a) - Q_{\min})/(Q_{\max} - Q_{\min})$ and $v(a) = 1 - (\log c(a) - \log c_{\min})/(\log c_{\max} - \log c_{\min})$, where $Q_{\min}, Q_{\max}$ and $c_{\min}, c_{\max}$ are shared quality and cost normalization bounds. For returned architecture set $\mathcal{R}$, we report

$$\mathrm{HVR}(\mathcal{R}) = \frac{\lambda_2\big(\bigcup_{a \in \mathcal{R}} [0, u(a)] \times [0, v(a)]\big)}{\mathrm{HV}(\mathcal{P}^\star)}. \tag{19}$$

Here, λ_2 denotes two-dimensional area, and the denominator is the reference-front hypervolume in the same normalized coordinates. Tabular experiments use the exact reference front; training-based experiments use the specified union-audit reference. Quality-floor costs at 90/95/97/99% provide complementary constrained operating points (Appendix C).

5 Experimental Setup

Tabular NAS benchmarks We combine NAS-Bench-101 (423,624 CIFAR-10 graphs) (Ying et al., 2019), NAS-Bench-201 (15,625 cells on CIFAR-10/100 and ImageNet16-120) (Dong & Yang, 2020), NATS-Bench topology/size spaces (Dong et al., 2022), seven-task TransNAS-Bench-101 Micro/Macro (Duan et al., 2021), and device-aware HW-NAS-Bench (Li et al., 2021). Matching released quality, proxy, and cost fields yields 77 task–cost conditions. Each method receives the same 256-candidate sample and a label budget $B \in \{4, 8, 16, 32\}$. Complete benchmark tables define the HVR normalization bounds and reference hypervolume. Comparisons include random selection, cost-only selection, individual proxies, fixed fusion, Online Hedge, fixed EHVI, full NICE-CV, and an evaluation-only oracle-best-single. To evaluate training-time allocation, we additionally use NAS-Bench-201 intermediate validation and epoch-time records to reconstruct 600 paired task–cost–seed runs. Promoting architecture a from fidelity $f_{\ell-1}$ to f_ℓ incurs the incremental cost $T(a, f_\ell) - T(a, f_{\ell-1})$, where $T(a, f)$ is its recorded cumulative training time. Baselines include random, GradNorm, NWOT, and fixed-fusion full-fidelity search, together with Random-SH, Early-SH, GradNorm-SH, NWOT-SH, and Fixed-SH.

Table 1: Training-based search protocols. Pool denotes the candidate count for pool-based methods; fraction is the training-data fraction, and cohort sizes specify the numbers trained at successive fidelities.

Task	Metric	Pool	Fraction	Batch	Fidelity	Cohort sizes
CIFAR-100	top-1	256	.25	256	1/3/9 epochs	32/8/4
Tiny-ImageNet	top-1	256	.25	256	1/3/9 epochs	32/8/4
Oxford-Pet	mIoU	128	1.0	4	.9/2.7/8.1k steps	32/8/4
Penn-Fudan	AP50	128	.50	1	.9/2.7/8.1k steps	32/8/4

Table 2: Tabular benchmark comparison: mean HVR over five NAS benchmark families.

Method	$B = 4$	$B = 8$	$B = 16$	$B = 32$
Random	.5857	.6714	.7496	.8116
Cost-only	.6075	.7306	.7965	.8260
Fixed fusion	**.7413**	.8160	.8557	.8835
Online Hedge	**.7413**	.8414	.8787	.8994
Fixed-EHVI	**.7413**	.8167	.8511	.8739
NICE-CV	**.7413**	**.8452**	**.8829**	**.9008**

Training-based search We evaluate classification on CIFAR-100 (Krizhevsky, 2009) and Tiny-ImageNet (Le & Yang, 2015), segmentation on Oxford-IIIT Pet (Parkhi et al., 2012), and detection on Penn-Fudan (Wang et al., 2007). The search space contains MobileNet-like backbones with discrete stem/stage widths, stage depths, MBConv kernel sizes, and expansion ratios. Task heads remain fixed: a linear classifier, a lightweight multi-scale decoder, or an FPN/RetinaNet-style detection head. Table 1 summarizes the training protocols. Classification uses fixed training/validation partitions and top-1 accuracy. Oxford-IIIT Pet uses an 80/20 split and trimap mIoU; Penn-Fudan uses a fixed 80/20 split and pedestrian AP50, with negative validation detection loss as the non-terminal signal. Optimizers and architectural ranges appear in Appendix C.

We compare two baseline groups. *Classical search baselines* include Random-SH (uniform sampling), NWOT-SH (activation-diversity ranking) (Mellor et al., 2021), Fixed-SH (GradNorm/NWOT fusion), Early-SH (validation-quality promotion), SH-EMOA (non-dominated selection using trained quality and latency), and TPE-HB (BOHB-style density-ratio proposals) (Falkner et al., 2018). *Recent NAS baselines* include AZ-NAS (Lee & Ham, 2024), MF-NAS (Phan & Luong, 2026), and R-MF-NAS (Phan & Luong, 2026), evaluated on Tiny-ImageNet, Oxford-IIIT Pet, and Penn-Fudan using reference-aligned implementations. These methods share the backbone family, task loss, split, fidelity schedule, and terminal metric; implementation details appear in Appendix C.5. Adaptive-MF is an ablation without the separately calibrated prior, while full NICE-CV corresponds to Adaptive-Calibrated.

Implementation and evaluation protocol Tabular experiments use 100 paired seeds per condition and training-based search comparisons use 20 paired seeds. Deployment latency is measured on RTX A6000 using synchronized batch-1 inference. Method-accounted search time charges every requested proxy computation, profiling operation, and training increment, including cached evaluations. We report paired mean differences with 95% bootstrap confidence intervals for the training-based search comparisons. Additional implementation and evaluation details are provided in the appendix.

6 Experimental Results

6.1 Search performance on tabular NAS benchmarks

Table 2 compares mean HVR across five complete tabular benchmark families, covering 77 task–cost conditions and 100 paired seeds per condition. Each method receives the same 256-candidate pool and a purchased-label budget $B \in \{4, 8, 16, 32\}$. The table includes random and cost-only selection, fixed proxy fusion, the online-weighting, acquisition ablations, and NICE-CV. At $B = 4$, fixed fusion, Online Hedge, Fixed-EHVI, and NICE-CV all obtain 0.7413 HVR because they share the initial

Table 3: Recorded NAS-Bench-201 comparison: mean HVR and method-accounted search time.

Method	HVR ↑	Recorded seconds ↓
Random full	.8574	70,784
GradNorm full	.9130	69,748
NWOT full	**.9456**	69,855
Fixed full	.9290	69,771
Random-SH	.7625	67,115
Early-SH	.8879	64,816
GradNorm-SH	.8887	63,758
NWOT-SH	.9098	64,664
Fixed-SH	.9080	65,860
NICE-CV	.9360	**62,056**

Table 4: Mean HVR and search time (minutes) in training-based searches.

	CIFAR-100		Tiny-ImageNet		Oxford-Pet		Penn-Fudan	
Method	HVR↑	Min.↓	HVR↑	Min.↓	HVR↑	Min.↓	HVR↑	Min.↓
A: Classical baselines and adaptive methods								
Random-SH	.3933	1.939	.3964	3.787	.5591	16.615	.4988	29.489
NWOT-SH	.6263	2.066	.6820	3.887	.5731	17.516	.5104	29.278
Fixed-SH	.6825	2.057	.7418	3.876	.7096	16.858	.7352	28.917
Early-SH	.7697	1.999	.8355	3.871	.8765	16.507	.7535	28.836
SH-EMOA	.5617	1.908	.6159	3.777	.7887	**16.127**	.7198	29.141
TPE-HB	.6394	**1.770**	.6727	3.733	.6410	16.366	.6510	**28.777**
Adaptive-MF	.7860	1.984	.8495	3.871	.8528	16.605	.7255	28.861
B: Recent NAS baselines								
AZ-NAS	–	–	.5662	**3.324**	.6397	18.100	.7000	30.164
MF-NAS	–	–	.1482	4.039	.0891	20.991	.1777	33.047
R-MF-NAS	–	–	.2161	4.015	.5364	18.865	.2179	32.496
NICE-CV	**.7906**	1.816	**.8808**	3.858	**.8813**	16.350	**.7724**	28.830

design and no validation comparison is yet available. This exceeds random selection (0.5857) and cost-only selection (0.6075). Once feedback becomes available, NICE-CV achieves the highest HVR among the search methods at every remaining budget: 0.8452, 0.8829, and 0.9008 at $B = 8, 16, 32$, respectively. At $B = 16$, NICE-CV improves on random selection by 0.1333 HVR and cost-only selection by 0.0864. Relative to fixed fusion, the paired improvement is 0.02720.

The ablation rows identify online calibration as the principal source of improvement. At $B = 16$, Online Hedge reaches 0.8787, compared with 0.8557 for fixed fusion and 0.8511 for Fixed-EHVI. Updating reliability contributes 0.0230 HVR over fixed fusion, while adding posterior EHVI to Online Hedge yields a further 0.00421 (95% CI $[0.00349, 0.00491]$). At $B = 32$, Online Hedge and full NICE-CV remain close (0.8994 versus 0.9008). These results show that learning proxy reliability supplies most of the observed gain, while the additional benefit of acquisition diminishes as more labels are purchased. Per-benchmark results appear in Appendix E, Tables 6–10. The aggregate advantage is relatively uniform across benchmarks.

Recorded-time NAS-Bench-201. Table 3 evaluates the full quality–search-time operating point. NICE-CV obtains 0.9360 HVR in 62,056 recorded seconds. Relative to fixed full-fidelity fusion it gains 0.00693 HVR (CI $[0.00482, 0.00907]$) and saves 10.78% time (CI $[10.40, 11.19]\%$). Relative to NWOT-SH it gains 0.02615 HVR and saves 4.13%. Full-fidelity NWOT produces 0.9456 HVR but consumes 69,855 seconds; NICE-CV recovers 99.0% of that HVR in 88.8% of its time. The result is exactly the operating regime targeted by Eq. equation 16: a small terminal-frontier sacrifice can be undesirable when considered alone yet superior as Pareto gain per second.

6.2 Training-based search performance across CV tasks

Table 4 jointly reports HVR and search time for classical search baselines and recent NAS baselines, alongside Adaptive-MF and NICE-CV. Panel A reports classical baselines and adaptive methods results; Panel B reports recent NAS baselines results. NICE-CV has the highest mean HVR on all four tasks compared to classical search baselines, adaptive method, and recent NAS baselines.

Against classical baselines, NICE-CV achieves the highest mean HVR on every task: 0.7906 on CIFAR-100, 0.8808 on Tiny-ImageNet, 0.8813 on Oxford-Pet, and 0.7724 on Penn-Fudan. The clearest gain over Early-SH occurs on Tiny-ImageNet: HVR increases from 0.8355 to 0.8808, with a paired mean gain of 0.04525 (95% CI $[0.01411, 0.08222]$). The gains over Fixed-SH and TPE-HB are 0.1390 and 0.2081, respectively. On CIFAR-100, NICE-CV reaches 0.7906 versus 0.6825 for Fixed-SH and 0.7697 for Early-SH. The corresponding Early-SH comparisons on Oxford-Pet and Penn-Fudan are 0.8813 versus 0.8765 and 0.7724 versus 0.7535. These three paired gains have confidence intervals containing zero: $[-0.05437, 0.09510]$, $[-0.01537, 0.02685]$, and $[-0.0476, 0.0854]$, respectively. Thus, the highest mean HVR on every task does not establish a significant advantage over Early-SH on every task. NICE-CV also exceeds AZ-NAS, MF-NAS, and R-MF-NAS on all three evaluated tasks. On Tiny-ImageNet, NICE-CV obtains 0.8808 HVR, followed by AZ-NAS at 0.5662, R-MF-NAS at 0.2161, and MF-NAS at 0.1482. The same ordering holds on Oxford-Pet, with 0.8813, 0.6397, 0.5364, and 0.0891, respectively. Penn-Fudan provides the same outcome: NICE-CV leads at 0.7248, followed by AZ-NAS at 0.7000, R-MF-NAS at 0.2179, and MF-NAS at 0.1777. R-MF-NAS improves on MF-NAS by 45.8% on Tiny-ImageNet, approximately sixfold on Oxford-Pet and 22.6% on Penn-Fudan, consistent with the benefit of trajectory diversification. AZ-NAS is stronger than both, but remains below NICE-CV on these tasks.

The search-time results show that NICE-CV combines strong frontier quality with competitive search speed. Although it is not the fastest method, the fastest baselines achieve lower HVR. On CIFAR-100, NICE-CV takes only 0.046 minutes longer than TPE-HB while improving HVR from 0.6394 to 0.7906. On Oxford-Pet and Penn-Fudan, it exceeds the fastest methods by only 0.223 and 0.053 minutes, respectively, while achieving the highest mean HVR. On Tiny-ImageNet, AZ-NAS is fastest at 3.324 minutes with relatively low HVR, whereas NICE-CV takes 3.858 minutes but achieve the best quality. Overall, NICE-CV maintains competitive search times while achieving the best mean HVR in the comparison, offering a favorable balance between search quality and computational cost.

Taken together, the HVR and timing results demonstrate a favorable quality–efficiency tradeoff for NICE-CV. Relative to the fastest baseline on each task, it improves HVR by 0.1512 on CIFAR-100, 0.2081 on Tiny-ImageNet, 0.0926 on Oxford-Pet, and 0.1214 on Penn-Fudan, with additional search times of only 0.046, 0.125, 0.223, and 0.053 minutes. Thus, online proxy calibration improves quality–latency frontier coverage while keeping search time close to the fastest classical strategies.

7 Conclusion

We presented NICE-CV, a multi-fidelity NAS method that treats zero-cost proxy reliability as an online search state. Rather than using proxy guidance only as a fixed initialization heuristic, NICE-CV continuously recalibrates proxy reliability from purchased validation evidence and uses the updated guidance to select subsequent architecture–fidelity actions. Measured inference latency is incorporated directly into the architecture objective, while expected hypervolume improvement per predicted incremental second determines how limited wall-clock training budget is allocated. Across complete tabular benchmarks, recorded multi-fidelity search, and training-based classification, segmentation, and detection tasks, NICE-CV consistently improves quality–latency frontier coverage under limited search budgets. The training-based results show that methods with lower search time generally recover lower-HVR architecture sets, whereas NICE-CV achieves the highest mean HVR on all four tasks with only modest additional search time relative to faster baselines. Controlled ablations further show that online proxy calibration accounts for most of the observed HVR improvement, supporting the central premise that proxy reliability should adapt as task-specific evidence accumulates. Overall, these results show that multi-fidelity NAS can benefit from closing the loop between cheap proxy guidance and validation feedback. By jointly deciding which proxies to trust, which architectures to evaluate, and how much training to allocate while accounting for measured inference latency and wall-clock search cost, NICE-CV provides a practical approach to discovering stronger quality–latency tradeoffs with less wasted training.

REFERENCES

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

Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*, 2019.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-All: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/pdf/1908.09791.pdf.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002. doi: 10.1109/4235.996017.

Xuanyi Dong and Yi Yang. NAS-Bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJxyZkBKDr.

Xuanyi Dong, Lu Liu, Katarzyna Musial, and Bogdan Gabrys. NATS-Bench: Benchmarking NAS algorithms for architecture topology and size. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3634–3646, 2022. doi: 10.1109/TPAMI.2021.3054824.

Yawen Duan, Xin Chen, Hang Xu, Zewei Chen, Xiaodan Liang, Tong Zhang, and Zhenguo Li. TransNAS-Bench-101: Improving transferability and generalizability of cross-task neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5251–5260, 2021.

Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1437–1446. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/falkner18a.html.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1675–1684, 2024.

Arjun Krishnakumar, Colin White, Arber Zela, Renbo Tu, Mahmoud Safari, and Frank Hutter. NAS-Bench-Suite-Zero: Accelerating research on zero cost proxies. *Advances in Neural Information Processing Systems*, 35:28037–28051, 2022.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Ya Le and Xuan Yang. Tiny ImageNet visual recognition challenge. Stanford CS231N course project, 2015. URL https://cs231n.stanford.edu/reports/2015/pdfs/yle_project.pdf.

Junghyup Lee and Bumsub Ham. AZ-NAS: Assembling zero-cost proxies for network architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5893–5903, 2024.

Chaojian Li, Zhongzhi Yu, Yonggan Fu, Yongan Zhang, Yang Zhao, Haoran You, Qixuan Yu, Yue Wang, and Yingyan (Celine) Lin. HW-NAS-Bench: Hardware-aware neural architecture search benchmark. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=_0kaDkv3dVf.

Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In Ryan P. Adams and Vibhav Gogate (eds.), *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pp. 367–377. PMLR, 2020. URL https://proceedings.mlr.press/v115/li20c.html.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018.

Marius Lindauer and Frank Hutter. Best practices for scientific research on neural architecture search. *Journal of Machine Learning Research*, 21(243):1–18, 2020.

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

Joe Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 7588–7598. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/mellor21a.html.

Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3498–3505, 2012.

Quan Minh Phan and Ngoc Hoang Luong. Robust and efficient multi-fidelity neural architecture search with zero-cost proxy-guided local search. *ACM Transactions on Evolutionary Learning and Optimization*, 6(2):1–31, 2026. doi: 10.1145/3774940.

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

Liming Wang, Jianbo Shi, Gang Song, and I-Fan Shen. Object detection combining recognition and segmentation. In Yasushi Yagi, Sing Bing Kang, In So Kweon, and Hongbin Zha (eds.), *Computer Vision – ACCV 2007*, volume 4843 of *Lecture Notes in Computer Science*, pp. 189–199. Springer, 2007. URL https://link.springer.com/chapter/10.1007/978-3-540-76386-4_17.

Ruochen Wang, Xiangning Chen, Minhao Cheng, Xiaocheng Tang, and Cho-Jui Hsieh. RANKNOSH: Efficient predictor-based architecture search via non-uniform successive halving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10377–10386, 2021.

Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10726–10734. IEEE, 2019.

Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: Stochastic neural architecture search. In *International Conference on Learning Representations*, 2019.

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

Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

Reproducibility Statement

We design all experiments to support deterministic replay and leakage-free comparison. The adaptive tabular, recorded multi-fidelity, training-based search, and final-training protocols are frozen separately before confirmation, with protocol hashes and evidence artifacts reported in Appendix C.

Tabular experiments use paired seeds and identical candidate pools across methods, while training-based comparisons use fixed data splits, fidelity schedules, and paired random seeds. Candidate generation, architecture encoding, and tie-breaking rules are deterministic.

Search-time accounting includes every method-specific proxy computation, deployment-latency profiling operation, and incremental training evaluation. Real-search latency is measured with synchronized batch-1 inference on the same RTX A6000 hardware class, and search-time and final latency measurements are recorded separately. Terminal audit quantities are evaluator-only and are never used to update proxy reliability, select architectures, or make fidelity decisions. Final evaluation is performed only after the search protocol, selected architectures, training schedule, and other evaluation settings have been fixed.

For every training-based run, the recorded artifacts include the task, architecture key, random seed, protocol hash, terminal audit values, measured latency, and method-accounted search time. Appendix C provides the executable protocol files, result summaries, and evidence-artifact paths needed to reconstruct the reported tables and statistical comparisons.

Statement on the Use of Generative AI

Generative AI tools were used to assist with language editing, document organization, and LaTeX drafting. The authors remain responsible for the scientific claims and verified the implementation descriptions, formulas, experimental records, and numerical results against the retained artifacts. AI tools were not treated as a source of experimental evidence.

A Proofs

A.1 Proof of Proposition 1

Let g be strictly increasing. For every pair a, b, $z(a) < z(b)$ iff $g(z(a)) < g(z(b))$, and equality is preserved. Every indicator in Eq. equation 6 is therefore unchanged; the same argument applies to Eq. equation 7. Pairwise losses depend only on signs of rank differences and validation order, so Eq. equation 8 and the complete weight trajectory are unchanged. Consequently the posterior predictions, EHVI values, and deterministic tie-broken actions are unchanged, provided wall-clock estimates themselves are not transformed. A merely non-decreasing transformation could introduce ties, hence strict monotonicity is required. □

A.2 Proof of Theorem 1

Write $L_{j,t} = \sum_{s=1}^{t} \ell_{j,s}$ and $W_t = \sum_j \exp(-\eta L_{j,t-1})$, so $w_{j,t} = \exp(-\eta L_{j,t-1})/W_t$. Hoeffding's lemma for a random variable in $[0, 1]$ gives

$$\log \frac{W_{t+1}}{W_t} = \log \sum_j w_{j,t} \exp(-\eta \ell_{j,t}) \tag{20}$$

$$\leq -\eta \langle \mathbf{w}_t, \boldsymbol{\ell}_t \rangle + \frac{\eta^2}{8}. \tag{21}$$

Summing over t yields

$$\log W_{T+1} - \log J \leq -\eta \sum_{t=1}^{T} \langle \mathbf{w}_t, \boldsymbol{\ell}_t \rangle + \frac{\eta^2 T}{8}. \tag{22}$$

For any expert j, $W_{T+1} \geq \exp(-\eta L_{j,T})$. Substitution, maximization over j, and division by η prove

$$\sum_t \langle \mathbf{w}_t, \boldsymbol{\ell}_t \rangle - \min_j L_{j,T} \leq \frac{\log J}{\eta} + \frac{\eta T}{8}. \tag{23}$$

Minimizing the right side at $\eta = \sqrt{8 \log J / T}$ gives $\sqrt{T \log J / 2}$. The fidelity-weighted loss in Eq. equation 8 remains in $[0, 1]$, so the same potential argument applies. □

A.3 Proof of Corollary 1

Taking the ratio of two generalized posterior weights in Eq. equation 10 cancels the normalizer:

$$\frac{w_{j,t+1}}{w_{h,t+1}} = \frac{p_0(j)}{p_0(h)} \exp\{-\eta(L_{j,t} - L_{h,t})\}. \tag{24}$$

Substituting $L_{j,t} - L_{h,t} \geq t\delta$ proves Eq. equation 18. □

A.4 Derivation of precision-weighted cross-fidelity fusion

Under conditional independence and zero-mean rank errors, the cross term in $\mathbb{E}[(\widetilde{q} - q)^2]$ vanishes, giving Eq. equation 12. Differentiation yields

$$\frac{\partial}{\partial \rho} \mathrm{MSE}(\widetilde{q}) = -2(1-\rho)\sigma_{p,j}^2 + 2\rho\sigma_f^2 = 0, \tag{25}$$

and hence Eq. equation 13. The second derivative $2(\sigma_{p,j}^2 + \sigma_f^2) > 0$ proves uniqueness. The resulting variance is smaller than either component variance, quantifying why proxy and partial-training evidence are complementary before validation becomes decisive.

Scope of the theorem. The result compares reliability loss with the best fixed proxy expert and makes no assumption that any proxy perfectly predicts final quality. It does not bound global Pareto regret because a finite pool may omit useful architectures, early observations may be biased, and a time model may be inaccurate.

B Algorithmic Specification

1. Compute deterministic proxy ranks $r_j(a)$ and deployment-efficiency ranks $e(a)$; initialize $w_{j,1} = 1/J$ and purchase the frozen space-filling design.
2. After each validation result, form the fidelity-weighted pairwise losses in Eq. equation 8 and apply the exponential update in Eq. equation 9.
3. Construct the cross-fidelity mixture, enumerate every legal start and promotion in Eq. equation 5, and evaluate Eq. equation 16 using incremental predicted time.
4. Execute the highest-rate affordable action, update budget and the time model, and repeat until no legal action fits.
5. Terminally audit the returned non-dominated set; audit values never enter a reliability or acquisition update.

Initial ranking costs $O(JN \log N)$ time and $O(JN)$ memory. With cached ranks, round t costs $O(Jt)$ for pairwise updates and $O(N_t(J + K_t))$ for exact two-dimensional acquisition with N_t legal actions and a K_t-point frontier.

C Extended Experimental Details

C.1 Claim-to-evidence map

Table 5 links the experimental results to their supporting artifacts, including aggregate summaries, per-seed records, search implementations, and evaluation protocols.

Table 5: Evidence artifacts supporting the experimental results.

Claim	Evidence artifact
Adaptive tabular sweep	`paper_experiments/results/adaptive_all_confirm_summary.json`
Adaptive protocol/code	`protocol_adaptive_v1.json`; `adaptive_pareto_search.py`
Recorded multi-fidelity	`results/multifidelity_confirm_summary.json`; `multifidelity_nb201.py`
Classification confirmation	`real_nas/results/v2_confirmation/*_seed*.json`; `v2_confirmation_summary.json`
Oxford-Pet confirmation	`results/oxford_pet_segmentation_confirmation_v1/*_seed*.json`; `summary.json`
Penn-Fudan confirmation	`results/penn_fudan_detection_confirmation_v1/*_seed*.json`; per-seed frozen outputs
AZ-NAS/MF-NAS/R-MF-NAS baselines	`official_reproduction_v1/real_external/results_v1/*_seed[01].json`; locked cross-task protocol
Final test-once runs	`real_nas/results/final_*_candidate*_seed*.json`; `final_train_summary.json`
Executable protocols	`real_nas/protocol_real_search_v1.json`; `protocol_final_train_v1.json`

C.2 Adaptive tabular accounting

The confirmation uses 100 seeds, 77 task–cost pairs, and four budgets, hence 7,700 paired search instances per budget. Every method receives the same 256-candidate pool. Random, cost-only, all compatible single proxies, fixed uniform fusion, online Hedge, fixed EHVI, and adaptive EHVI are evaluated without loading test fields. NAS-Bench-101 uses path count and operation-weighted path mass; the remaining tables use complete released GradNorm, NWOT, or Zen records.

At budget four, the four adaptive variants share the same frozen initial design and must coincide. This is both a design consequence and a pipeline check. Adaptation starts only after purchased feedback exists. Confidence intervals in the main text pair the same task, cost, pool, and seed before aggregation.

C.3 Multi-fidelity accounting

The recorded NAS-Bench-201 experiment crosses three datasets with parameter and FLOP objectives. Each method uses an equal total-time allowance; completing fewer full-fidelity candidates is therefore permitted. Full-fidelity methods spend approximately 69–71k recorded seconds, while adaptive multi-fidelity spends 62.1k and completes approximately eight promoted candidates. The comparison to full NWOT is explicitly non-dominating: adaptive saves time but gives up 0.00961 HVR.

C.4 Real architecture family and timing

The family comprises four inverted-bottleneck stages. Stem width, stage width, depth, kernel size, and expansion ratio are discrete. Candidate keys are hashes of the complete configuration. Search latency and final latency are independently re-measured on the same A6000 class; final timing uses more warmups and repetitions, so small numerical differences are expected. The pool-based methods in Table 4, Panel A, see exactly the same generated pool for a seed: 256 candidates for classification and 128 for segmentation/detection. Only 32 candidates receive first-rung training, followed by 8 and 4 promotions.

Classification search uses a quarter of the training split to control cost. Segmentation uses the fixed Oxford-Pet train partition, while Penn-Fudan uses a fixed development split and fractional sampling specified before confirmation. Final classification training restores the complete official training split and never uses the official test/validation split until the fixed terminal epoch. Output JSON records task, architecture key, seed, protocol hash, synthetic/test-loading flags, terminal audit, latency, and method-accounted seconds.

C.5 AZ-NAS, MF-NAS, AND R-MF-NAS BASELINE IMPLEMENTATIONS

The formal baseline evaluation preserves the defining search mechanism of each method while replacing the task backend. MF-NAS and R-MF-NAS use the fixed-length encoding

$$g(a) = (s, w_1, \ldots, w_4, d_1, \ldots, d_4, k_{1,1}, \ldots, k_{4,3}, e_{1,1}, \ldots, e_{4,3}) \tag{26}$$

with latent genes retained for inactive blocks. A one-move neighbor changes one coordinate to another legal value. First-improvement local search tests randomized one-move neighborhoods and escapes a local optimum through a random valid distance-two move. Exactly 2,000 distinct architecture hashes are charged. SynFlow uses absolute parameter linearization, an all-ones input, double precision, and the sum of $|\theta \odot \nabla_\theta \mathbf{1}^\top f_\theta(\mathbf{1})|$. MF-NAS takes the 32 highest scores. R-MF-NAS takes $\lceil 32/3 \rceil$ top scores and fills the cohort from unique start and middle trajectory states before SH.

The AZ-NAS baseline implements the published four components. For layer features $X_l \in \mathbb{R}^{n_l \times c_l}$, expressivity and progressivity are

$$E(a) = \sum_l H\left(\operatorname{eig}\left(\frac{(X_l - \bar{X}_l)^\top (X_l - \bar{X}_l)}{n_l}\right)\right), \tag{27}$$

$$P(a) = \min_l \big(H_{l+1} - H_l\big), \tag{28}$$

where eigenvalues are normalized to a probability vector before entropy. Trainability uses the largest singular value σ_l of the adjacent-layer random-sign gradient cross-covariance,

$$T(a) = \frac{1}{L-1} \sum_{l=1}^{L-1} \left(-\sigma_l - \frac{1}{\sigma_l + 10^{-6}} + 2\right), \tag{29}$$

with PixelUnshuffle matching unequal spatial resolutions. Complexity $C(a)$ is measured MACs. The nonlinear aggregation is

$$S_{\mathrm{AZ}}(a) = \sum_{g \in \{E,P,T,C\}} \log\left(\frac{\operatorname{rank}_g(a)}{|\mathcal{P}|}\right). \tag{30}$$

Tiny-ImageNet uses native 64^2 proxy inputs; dense-task SynFlow uses canonical 224^2 backbone inputs, while AZ-NAS uses the native 320^2 segmentation and 640^2 detection resolution.

D BENCHMARK UNIVERSE AND GENERALIZATION

Complete tabular universes permit exact validation fronts and HVR denominators within each published table. Real experiments replace that finite oracle with actual training, independently generated candidate pools, and a union terminal audit. The two levels answer complementary questions: tables identify mechanism under exact counterfactuals; real tasks test whether the mechanism survives optimization noise, heterogeneous heads, and measured device latency.

E PER-BENCHMARK TABULAR RESULTS

Tables 6–10 report confirmation results for NAS-Bench-101, NAS-Bench-201, TransNAS-Bench-101 Macro and Micro, and HW-NAS-Bench. Each table averages HVR over its task–cost conditions and 100 paired seeds per condition.

Table 6: NAS-Bench-101 confirmation results.

Method	$B = 4$	$B = 8$	$B = 16$	$B = 32$
Random	.5663	.6208	.6554	.6994
Cost-only	.7751	.7842	.7929	.7997
Fixed fusion	**.7837**	.7920	.7983	.8035
Online Hedge	**.7837**	**.7931**	.7983	.8026
Fixed-EHVI	**.7837**	.7924	.7983	.8032
NICE-CV	**.7837**	.7929	**.7990**	**.8036**

Table 7: NAS-Bench-201 confirmation results.

Method	$B = 4$	$B = 8$	$B = 16$	$B = 32$
Random	.6283	.7539	.8427	.8912
Cost-only	.7524	.7859	.8576	.8862
Fixed fusion	**.8325**	.9005	.9302	.9460
Online Hedge	**.8325**	**.9093**	.9346	.9452
Fixed-EHVI	**.8325**	.8949	.9227	.9386
NICE-CV	**.8325**	.9075	**.9391**	**.9502**

Table 8: TransNAS-Bench-101 Macro confirmation results.

Method	$B = 4$	$B = 8$	$B = 16$	$B = 32$
Random	.6071	.6870	.7588	.8088
Cost-only	.5935	.6352	.6818	.7382
Fixed fusion	**.7030**	.7560	.8002	.8399
Online Hedge	**.7030**	.7667	.8193	.8622
Fixed-EHVI	**.7030**	.7424	.7865	.8270
NICE-CV	**.7030**	**.7894**	**.8310**	**.8628**

Table 9: TransNAS-Bench-101 Micro confirmation results.

Method	$B = 4$	$B = 8$	$B = 16$	$B = 32$
Random	.6022	.6865	.7484	.8153
Cost-only	.6558	.7388	.7899	.8062
Fixed fusion	**.7401**	.8279	.8739	.8984
Online Hedge	**.7401**	**.8426**	**.8816**	**.9003**
Fixed-EHVI	**.7401**	.8309	.8707	.8915
NICE-CV	**.7401**	**.8426**	**.8816**	**.9003**

Table 10: HW-NAS-Bench confirmation results.

Method	$B = 4$	$B = 8$	$B = 16$	$B = 32$
Random	.5482	.6330	.7298	.8014
Cost-only	.5363	.7820	.8776	.8983
Fixed fusion	**.7487**	.8365	.8724	.8980
Online Hedge	**.7487**	**.8871**	.9162	.9246
Fixed-EHVI	**.7487**	.8478	.8740	.8875
NICE-CV	**.7487**	.8805	**.9180**	**.9270**

F Derivation of the acquisition objective

For a two-objective frontier sorted so $u_1 \leq \cdots \leq u_K$ and $v_1 \geq \cdots \geq v_K$, define $v_0 = 0$. The dominated hypervolume relative to the origin is

$$\mathrm{HV}(\mathcal{H}) = \sum_{k=1}^{K} (u_k - u_{k-1}) v_k, \qquad u_0 = 0. \tag{31}$$

For candidate efficiency e, only rectangles not already dominated contribute. If Q has posterior CDF $F_{a,t}$ in normalized coordinates, Fubini's theorem gives

$$\mathrm{EHVI}_t(a) = \int_0^1 \big[1 - F_{a,t}(q)\big] \Big(e(a) - v_{\mathcal{H}_t}(q)\Big)_+ \, dq, \tag{32}$$

where $v_{\mathcal{H}_t}(q)$ is the best efficiency already attained at quality at least q. This expression clarifies three design choices: uncertainty is integrated rather than replaced by a posterior mean, improvements already covered by the frontier contribute zero, and a low-latency candidate may be valuable even when its expected quality is not maximal. Dividing by incremental $\widehat{\tau}(a, f)$ converts geometric value to a rate and makes promotion comparable with starting a new candidate.

G Complexity and reproducibility checklist

Ranking J proxies over N candidates costs $O(JN \log N)$ and $O(JN)$ memory. With cached ranks, a direct online update at observation t costs $O(Jt)$; evaluating all legal actions costs $O(NJ + NK)$ for a K-point frontier. The real pools are small enough to enumerate actions exactly. Deterministic candidate generation, paired seeds, frozen tie-breaking, protocol hashes, evaluator-only terminal audits, and explicit method-time accounting make every table reproducible from its JSON artifacts. The official ICLR 2027 style is used.