ELAX: LLM-GUIDED EVOLUTIONARY NEURAL ARCHITECTURE SEARCH FOR HETEROGENEOUS EDGE SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Deploying neural networks across heterogeneous edge platforms is challenging because architecture accuracy–latency trade-offs vary across hardware, while independently training and profiling large numbers of candidate architectures is expensive. We present ELAX, an LLM-assisted evolutionary neural architecture search framework that combines elastic weight sharing with hardware-aware multi-objective optimization. ELAX converts a pretrained backbone into an elastic supernet and trains its shared weights using feature-level transfer and mixed random and LLM-guided subnet sampling. The trained weights are then frozen and reused during profile-specific evolutionary search. Rather than serving as a fitness predictor, the LLM proposes legal subnet configurations and local architectural mutations conditioned on the current architecture, observed performance, and hardware constraints. We evaluate ELAX on elastic CNN and transformer backbones across CIFAR-10, CIFAR-100, and ImageNet-1K, and perform hardware-aware benchmark comparisons on NAS-Bench-201 and HW-NAS-Bench across multiple device profiles. The results show that ELAX finds subnetworks with varying accuracy and latency trade-offs across devices and achieves competitive Pareto-front quality under matched architecture-evaluation budgets compared with adapted multi-objective search baselines.

1 INTRODUCTION

Neural Architecture Search (NAS) automates neural network design by exploring candidate structures, such as operator types, connectivity, depth, and width, to identify architectures with favorable predictive performance and computational efficiency (Zoph & Le, 2017; Liu et al., 2019). Despite its success, NAS remains computationally expensive because an effective search typically requires evaluating a large number of candidate architectures. This challenge is particularly important for hardware-aware NAS, where architecture quality depends not only on accuracy but also on *device-specific* constraints such as latency, memory consumption, and energy usage. In heterogeneous edge environments, an architecture that performs efficiently on a cloud GPU may be too slow or memory-intensive for a mobile processor, embedded GPU, or accelerator. Consequently, architecture search may need to be repeated for different hardware profiles and resource budgets, making scalable edge adaptation costly (Liu et al., 2026; Lyu et al., 2022; Thind et al., 2026). Figure 1 illustrates this heterogeneity: under the same search space and evaluation budget, accuracy–latency frontiers differ across five HW-NAS-Bench profiles (Li et al., 2021), and an architecture feasible on one device may violate another's latency constraint.

Weight-sharing NAS and elastic networks reduce this overhead by encoding many candidate architectures within a single shared-weight *supernet* (Guo et al., 2020; Jeon et al., 2025; Li et al., 2025; You et al., 2020; Cai et al., 2020). A candidate *subnet* selects a specific combination of layers, widths, blocks, or operators from the supernet and inherits the corresponding trained weights. Candidate architectures can therefore be evaluated without being independently trained from scratch (Guo et al., 2020; Cai et al., 2020). Once-for-All (OFA) (Cai et al., 2020), for example, separates one-time supernet training from hardware-specific subnet selection, allowing the training cost to be amortized across multiple deployment targets. However, identifying high-quality architectures within the resulting combinatorial subnet space remains a nontrivial optimization problem.

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.

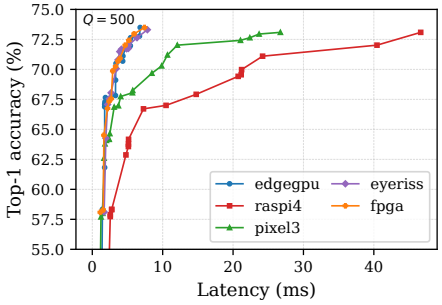


Figure 1: Accuracy–latency Pareto frontiers across HW-NAS-Bench heterogeneous hardware profiles on NAS-Bench-201.

To address this challenge, evolutionary NAS has been widely explored for discrete architecture optimization (Real et al., 2019). It iteratively improves a population of candidate architectures through fitness-based selection, mutation, and crossover. Multi-objective evolutionary search is particularly suitable for hardware-aware NAS because it can preserve solutions representing different trade-offs between predictive performance and computational cost (Xue et al., 2023). Recent methods have improved search efficiency through faster evolutionary exploration and architecture-specific surrogate evaluation (Cai et al., 2024; Lin & Luo, 2025). Nevertheless, conventional methods commonly generate candidates using random or predefined mutation operators rather than jointly conditioning their proposals on the current architecture, observed performance, target hardware, and accumulated search history. Consequently, a substantial portion of the evaluation budget may be spent on unpromising architectural modifications. Complementing evolutionary search, multi-objective Bayesian optimization offers another sample-efficient approach for costly architecture search. qLogEHVI selects candidate batches that maximize expected hypervolume improvement, while LaMOO uses search-space partitioning and MOBO-OSD promotes diverse exploration through orthogonal search directions (Ament et al., 2023; Zhao et al., 2024; Ngo et al., 2025). However, these methods rely on surrogate-based acquisition optimization rather than using the current architecture and target hardware constraints to guide local architecture modifications.

Recent progress in Large Language Models (LLMs) introduces a new opportunity for architecture search (Nasir et al., 2024; Zheng et al., 2023; Altman et al., 2024; Wang et al., 2023). LLMs can process structured architecture descriptions, reason about the functional and computational roles of neural operators, and generate modifications conditioned on textual objectives. Existing LLM-assisted NAS approaches have mainly used language models to generate complete architectures, produce search programs, or iteratively refine candidate designs (Nasir et al., 2024; Altman et al., 2024). However, treating the LLM only as a one-shot architecture generator does not fully exploit its potential in constrained edge deployment. Edge NAS requires iterative decisions: *whether to increase capacity, remove a costly operator, preserve accuracy-critical blocks, or bias search toward a device-specific latency budget*. These decisions naturally fit an evolutionary search process in which the LLM can guide mutations while population-based selection preserves empirical feedback.

In this paper, we present ELAX, an LLM-guided *evolutionary* NAS framework for hardware-efficient edge AI deployment. ELAX first adapts a pretrained backbone into an elastic supernet, allowing candidate subnets to inherit shared weights and be evaluated without independent retraining. The trained weights are then frozen, and multi-objective evolutionary search is performed for each target hardware profile. Within the search loop, an LLM-based mutation controller uses the current architecture, hardware profile, and latency constraint to rank structured local edits. A deterministic validator rejects malformed or illegal proposals and applies a legal random mutation when necessary. All candidates are evaluated and selected using the same accuracy and latency objectives, and the resulting Pareto archive supports subnet selection across latency budgets. This design combines LLM-guided candidate generation with robust population-based evolutionary optimization.

This paper makes the following contributions:

- We propose ELAX, an LLM-guided evolutionary NAS framework that combines hardware-aware multi-objective search with a two-stage shared-weight training procedure for constructing elastic supernets from pretrained backbones.
- We design an LLM-based mutation controller that proposes valid hardware-aware edits to elastic subnets and NAS-cell operators, combining language-model guidance with evolutionary exploration while retaining objective candidate evaluation and selection.
- We evaluate ELAX on image classification benchmarks, elastic convolutional and transformer backbones, and heterogeneous hardware profiles, comparing accuracy–latency trade-offs with unguided search and multi-objective optimization baselines under comparable budgets.

2 Related Work

Efficient and hardware-aware NAS. Neural Architecture Search (NAS) automates model design under accuracy and efficiency constraints, but early reinforcement-learning and evolutionary approaches incurred substantial search costs (Zoph & Le, 2017; Real et al., 2023). Differentiable and one-shot methods reduce this cost through continuous relaxation or weight-sharing supernets (Liu et al., 2019; Guo et al., 2020; You et al., 2020). Subsequent work improves inherited-weight ranking through fair sampling and subnet-aware optimization (Chu et al., 2021; Wang et al., 2021; Jeon et al., 2025), while training-free indicators, knowledge distillation, and architecture-specific evaluation improve search efficiency and subnet quality (Jiang et al., 2023; Lee & Ham, 2024; Lin & Luo, 2025; Peng et al., 2020; Li et al., 2020; Nath et al., 2024; Xue et al., 2025). OFA supports elastic depth, width, kernel size, and resolution (Cai et al., 2020), with later methods extending elastic training to multi-objective optimization and mobile transformers (Girard et al., 2024; Tang et al., 2023). Nevertheless, selecting strong candidates from the resulting combinatorial subnet spaces remains challenging.

Hardware-aware NAS incorporates deployment objectives such as latency, energy, and memory into architecture optimization (Cai et al., 2019; Wu et al., 2019; Gao et al., 2024). Complementary approaches improve edge-efficient transformers through structured or token-level pruning, convolution–transformer hybrids, and joint model pruning and cloud–device partitioning (Yu & Xiang, 2023; Wang et al., 2024; Zheng, 2025; Zeng et al., 2025; Jiang et al., 2025). These studies highlight the need to adapt architectures to heterogeneous hardware and network constraints.

LLM-Guided architecture search. Recently, Large Language Models (LLMs) have been explored as architecture generators and search controllers (Nasir et al., 2024; Zheng et al., 2023; Altman et al., 2024; Wang et al., 2023). LLMatic combines code-generating LLMs with quality-diversity optimization (Nasir et al., 2024), while subsequent approaches incorporate feedback, reflection, ranking, and collaborative or multi-agent search (Cai et al., 2025; Ji et al., 2025; Hu et al., 2025; Li et al., 2026; Zhu et al., 2025). Related generative and predictor-based methods learn architecture priors to reduce search effort (Yu et al., 2026; Lv et al., 2026). However, LLM-generated candidates still require external validation and empirical evaluation, and few existing methods jointly integrate LLM-guided mutation, elastic weight sharing, and hardware-aware optimization. In contrast, ELAX uses the LLM as a constrained controller in two complementary roles: guiding subnet sampling during shared-weight training and proposing hardware-conditioned local mutations during evolutionary search.

3 Problem Formulation

We consider a pretrained backbone $\mathcal{N}_0$ and a task dataset $\mathcal{D} = \{\mathcal{D}_{\mathrm{tr}}, \mathcal{D}_{\mathrm{val}}\}$, where $\mathcal{D}_{\mathrm{tr}}$ and $\mathcal{D}_{\mathrm{val}}$ denote the training and validation sets, respectively. The pretrained backbone is elasticized into a weight-sharing supernet $\mathcal{S}(\mathbf{W})$ with shared parameters $\mathbf{W}$ and a finite architecture space $\mathcal{A}$. Let the searchable architecture contain n discrete decisions. A candidate architecture $\mathbf{a}$ is encoded as

$$\mathbf{a} = (a_1, \ldots, a_n) \in \mathcal{A} = \mathcal{O}_1 \times \cdots \times \mathcal{O}_n, \tag{1}$$

where a_i is the selected option for the i-th architectural decision, and $\mathcal{O}_i$ is its set of allowable choices. Each candidate architecture $\mathbf{a} \in \mathcal{A}$ activates a subset of the shared parameters, denoted by $\mathbf{W}_{\mathbf{a}} \subseteq \mathbf{W}$, and forms an executable subnet.

We consider a set of heterogeneous hardware profiles $\mathcal{H}_{\mathrm{hw}} = \{h_1, \ldots, h_K\}$. For a candidate architecture $\mathbf{a}$ and hardware profile h, let $\mathrm{Acc}(\mathbf{a})$ denote its validation accuracy and $\mathrm{Lat}_h(\mathbf{a})$ denote its inference latency. The latency of the same architecture may vary substantially across hardware profiles because different devices exhibit different operator execution characteristics.

Given a target hardware profile h and the maximum tolerable inference latency τ_h, the hardware-aware architecture search objective is

$$\mathbf{a}^*_{h,\tau_h} \in \arg\max_{\mathbf{a}\in\mathcal{A}} \mathrm{Acc}(\mathbf{a}), \quad \text{s.t.} \quad \mathrm{Lat}_h(\mathbf{a}) \le \tau_h, \tag{2}$$

where $\mathbf{a}^*_{h,\tau_h}$ denotes an optimal subnet for hardware profile h under latency SLO τ_h. Solving the objective function independently for every latency budget would repeat much of the same search. We

therefore formulate architecture search as the following hardware-specific bi-objective optimization problem:

$$\min_{\mathbf{a}\in\mathcal{A}} \mathbf{g}_h(\mathbf{a}) = (-\,\mathrm{Acc}(\mathbf{a}), \mathrm{Lat}_h(\mathbf{a}))\,. \tag{3}$$

where $\mathbf{g}_h(\mathbf{a})$ denotes the objective vector of architecture $\mathbf{a}$ under hardware profile h.

Solving this problem presents challenges. First, the architecture space $\mathcal{A}$ grows combinatorially with the number of architectural decisions, making exhaustive exploration impractical. Second, architecture latency and even relative latency rankings vary across hardware profiles, requiring the search policy to adapt its exploration to the target device. These challenges motivate the design of ELAX, which leverages shared-weight training to amortize candidate optimization, employs a profile-conditioned LLM controller to propose architecture mutations, and performs hardware-aware evolutionary search under a finite architecture-evaluation budget.

4 ELAX DESIGN

4.1 OVERVIEW

ELAX combines elastic weight sharing with LLM-guided multi-objective evolutionary search. Its central design principle is to use an LLM as a constrained mutation proposer rather than as an architecture generator, fitness predictor, or final decision maker. Each LLM-proposed edit is restricted to the predefined architecture space, validated deterministically, and evaluated using the same accuracy–latency objectives as candidates generated through random mutation and crossover. As illustrated in Fig. 2, the ELAX workflow comprises two phases connected by a shared LLM-guided architecture controller. Phase 1 is performed once for each backbone–task pair. It converts a pretrained backbone into an elastic supernet and optimizes the shared parameters through feature-level transfer and task-level fine-tuning across sampled subnets. During training, the LLM controller selectively proposes legal subnet configurations, while random sampling preserves broad coverage of the architecture space. Phase 2 freezes the trained supernet weights and performs profile-specific evolutionary Pareto search. The search generates candidates through crossover, random mutation, and selective LLM-guided mutation, evaluates each unique candidate using validation accuracy and hardware-specific latency, and updates a non-dominated archive. After the evaluation budget is exhausted, ELAX returns the most accurate architecture satisfying the target latency budget.

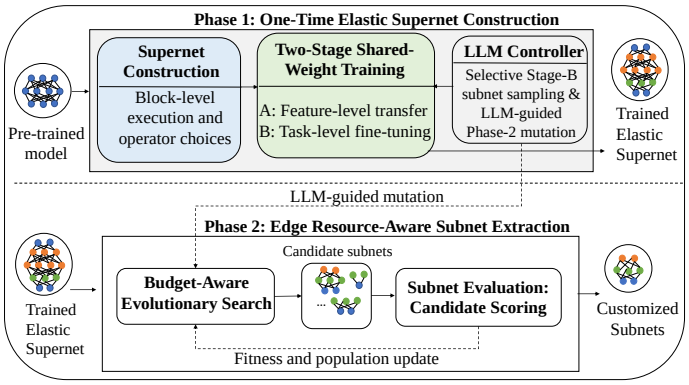


Figure 2: Overview of ELAX. Following two-stage elastic-supernet training, selective LLM-guided mutation and evolutionary search identify Pareto-optimal subnets under profile-specific latency constraints.

4.2 PHASE 1: ELASTIC SUPERNET CONSTRUCTION

4.2.1 SUPERNET CONSTRUCTION

Given the architecture space $\mathcal{A}$, we convert the pretrained backbone $\mathcal{N}_0$ into an elastic supernet $\mathcal{S}(\mathbf{W})$. For an architecture $\mathbf{a} \in \mathcal{A}$, the corresponding subnet activates parameters $\mathbf{W}_\mathbf{a} \subseteq \mathbf{W}$ and computes its forward function as:

$$f_\mathbf{a}(\mathbf{x}) = f(\mathbf{x}; \mathbf{W}_\mathbf{a}, \mathbf{a}), \tag{4}$$

where $\mathbf{x}$ is an input sample. Each architectural decision selects a registered block, branch, operator, or shape-compatible bypass, ensuring that every encoding in $\mathcal{A}$ maps to an executable subnet. The

supernet is initialized from $\mathcal{N}_0$ and optimized by sampling different execution paths across mini-batches, thereby sharing training across candidate architectures.

4.2.2 TWO-STAGE SHARED-WEIGHT TRAINING

We optimize the supernet in two stages. Stage A transfers intermediate representations from a fixed pretrained teacher network to sampled student subnets, while Stage B fine-tunes the shared parameters across diverse subnet configurations for the target task.

Stage A: *Feature-level knowledge transfer.* Let f_T denote the teacher network and $f_{\mathbf{a}}$ denote the subnet specified by $\mathbf{a}$. For an input $\mathbf{x}$, let $\mathbf{z}_j^T(\mathbf{x})$ and $\mathbf{z}_j^S(\mathbf{x};\mathbf{a})$ denote their feature tensors at location j. The set $\mathcal{J}(\mathbf{a})$ contains the locations at which the corresponding teacher and student features have compatible dimensions. The feature-transfer loss $\mathcal{L}_{\text{feat}}$ is

$$\mathcal{L}_{\text{feat}}(\mathbf{a}) = \frac{1}{|\mathcal{J}(\mathbf{a})|} \sum_{j \in \mathcal{J}(\mathbf{a})} \left\| \text{norm}(\mathbf{z}_j^S) - \text{norm}(\mathbf{z}_j^T) \right\|_2^2 . \tag{5}$$

Let y be the ground-truth class label and $\mathcal{L}_{\text{CE}}(f_{\mathbf{a}}(\mathbf{x}), y)$ denote the cross-entropy classification loss. The Stage-A objective is

$$\mathcal{L}_A = \begin{cases} \lambda_f \mathcal{L}_{\text{feat}} + \lambda_c \mathcal{L}_{\text{CE}}, & |\mathcal{J}(\mathbf{a})| > 0, \\ \mathcal{L}_{\text{CE}}, & |\mathcal{J}(\mathbf{a})| = 0, \end{cases} \tag{6}$$

where λ_f and λ_c control the contributions of feature transfer and task supervision. When compatible feature locations are unavailable, Stage A becomes classification-based warmup.

Stage B: *LLM-guided shared-weight task fine-tuning.* Stage B adapts the shared parameters to the target task by jointly optimizing over training samples and subnet configurations. Let q_{tr} denote the distribution used to sample architectures during training. The training objective $\mathcal{L}_B(\mathbf{W})$ is

$$\mathcal{L}_B(\mathbf{W}) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_{\text{tr}}} \mathbb{E}_{\mathbf{a}\sim q_{\text{tr}}} \left[\mathcal{L}_{\text{CE}}\big(f(\mathbf{x}; \mathbf{W}_{\mathbf{a}}, \mathbf{a}), y\big) \right]. \tag{7}$$

The outer expectation averages the loss over the training distribution, whereas the inner expectation accounts for architectures sampled from q_{tr}. Because different architectures activate different subsets of $\mathbf{W}$, minimizing this objective trains the shared parameters across the execution paths considered during subsequent subnet search. The optimized shared weights are obtained as

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} \mathcal{L}_B(\mathbf{W}). \tag{8}$$

The subnet-sampling distribution combines random exploration with validated LLM-guided sampling:

$$q_{\text{tr}} = (1 - \rho_{\text{tr}}) q_{\text{rand}} + \rho_{\text{tr}} q_{\text{LLM}}, \qquad 0 \le \rho_{\text{tr}} \le 1, \tag{9}$$

where q_{rand} denotes the random-sampling distribution, q_{LLM} is induced by legal subnet configurations proposed by the LLM controller, and ρ_{tr} controls the probability of guided sampling. Random sampling maintains broad coverage of the architecture space, whereas LLM-guided sampling increases exposure to potentially promising execution paths. Overall, Stage B trains the shared weights across diverse subnets before the Phase 2 evolutionary architecture search. The optimized weights $\mathbf{W}^*$ are then frozen so that all candidate architectures are evaluated under a common weight state during subnet search.

4.3 LLM-GUIDED CONTROLLER

For a selected parent architecture $\mathbf{a}$, the controller forms a context $\mathbf{c}$ from its serialized architectural decisions, estimated accuracy and latency, target hardware profile h, latency bound τ_h, a summary of the current non-dominated archive $\mathcal{R}$, and the allowable choices $\{\mathcal{O}_i\}_{i=1}^n$. The archive summary reports the highest accuracy and lowest latency observed so far. When constructing a complete Pareto frontier without a fixed latency bound, the parent latency is used as the local target in the prompt. Given $\mathbf{c}$, the controller returns a structured local edit $\Delta\mathbf{a}$,

$$\Delta\mathbf{a} = (i, a_i, \widetilde{a}_i, \gamma), \tag{10}$$

where i identifies the architectural decision to modify, a_i is its current value, $\widetilde{a}_i$ is the proposed replacement, and γ is the controller-reported confidence.

Let $\mathcal{V}_{\mathcal{A}}(\Delta\mathbf{a}, \mathbf{a})$ denote a deterministic validator. An edit is accepted only if it follows the required schema: its replacement belongs to the corresponding choice set, the replacement differs from the current choice, and its confidence satisfies the configured acceptance criterion. The guided mutation is

$$\mathrm{Mut}_{\mathrm{guide}}(\mathbf{a}; \mathbf{c}) = \begin{cases} \mathrm{Apply}(\mathbf{a}, \Delta\mathbf{a}), & \mathcal{V}_{\mathcal{A}}(\Delta\mathbf{a}, \mathbf{a}) = 1, \\ \mathrm{Mut}_{\mathrm{rand}}(\mathbf{a}), & \text{otherwise}, \end{cases} \tag{11}$$

where Apply applies the accepted edit and $\mathrm{Mut}_{\mathrm{rand}}$ performs a random mutation from the same architecture space. Therefore, missing, malformed, illegal, or rejected controller outputs cannot interrupt the search. To limit LLM overhead and preserve unguided exploration, the controller is invoked only for a fraction of offspring. Given parents $\mathbf{a}$ and $\mathbf{b}$, the offspring is generated as

$$\mathbf{a}' = \begin{cases} \mathrm{Mut}_{\mathrm{guide}}(\mathbf{a}; \mathbf{c}), & u < \rho_{\mathrm{LLM}}, \\ \mathrm{CrossMut}(\mathbf{a}, \mathbf{b}), & u \geq \rho_{\mathrm{LLM}}, \end{cases} \qquad u \sim \mathrm{Uniform}(0, 1), \tag{12}$$

where u is a uniform random draw, ρ_{LLM} is the guidance probability, and $\mathrm{CrossMut}$ denotes unguided evolutionary variation using crossover and random mutation. Every offspring is evaluated using the same accuracy–latency criteria and enters the same Pareto selection procedure.

4.4 Phase 2: Edge Resource-Aware Subnet Extraction

Candidate evaluation. At each search iteration, the evolutionary procedure generates candidate subnets, evaluates them using the profile-specific evaluator $\mathcal{E}_h$, and then updates the population. The evaluator scores each candidate $\mathbf{a}$ in terms of validation accuracy and latency, i.e., $\mathcal{E}_h(\mathbf{a}) = (\mathrm{Acc}(\mathbf{a}), \mathrm{Lat}_h(\mathbf{a}))$. For elastic subnets, accuracy is computed using weights inherited from the frozen supernet, avoiding the need to retrain each candidate independently. The resulting values instantiate the bi-objective formulation in Eq. (3).

Evolutionary Pareto search. We optimize the bi-objective formulation in Eq.(3) using NSGA-II (Deb et al., 2002). The initial population $\mathcal{P}_0$ contains P distinct candidates. Each generation uses binary tournament selection based on non-dominated rank and crowding distance. Offspring are generated through either the LLM-guided mutation in Eq. (12) or conventional crossover and random mutation. Environmental selection then retains P candidates, balancing objective quality and diversity along the estimated frontier. Let $\mathcal{C}_t$ be the offspring generated at iteration t and let $\mathcal{R}_t$ be the running non-dominated archive. The archive update is

$$\mathcal{R}_t = \mathrm{ND}\left(\mathcal{R}_{t-1} \cup \mathcal{P}_t \cup \mathcal{C}_t\right), \tag{13}$$

where $\mathrm{ND}(\cdot)$ returns the non-dominated subset. Maintaining a running archive prevents previously discovered frontier solutions from being lost during population replacement.

The architecture-evaluation budget Q counts distinct candidate evaluations. An evaluated-set cache stores every queried architecture; duplicate offspring are regenerated without consuming the budget. Thus, Q measures the number of unique points sampled from the architecture space. After Q distinct architecture evaluations, we obtain the final non-dominated archive $\mathcal{R}_T$. Its subset satisfying the requested latency bound is

$$\mathcal{R}_{h,\tau_h} = \left\{\mathbf{a} \in \mathcal{R}_T : \mathrm{Lat}_h(\mathbf{a}) \leq \tau_h\right\}, \tag{14}$$

where $\mathcal{R}_{h,\tau_h}$ denotes the feasible subset of the final non-dominated archive $\mathcal{R}_T$, containing the architectures discovered by the search that satisfy latency τ_h under profile h.

5 Evaluation

We evaluate ELAX along three dimensions: (i) whether the proposed shared-weight training supports elastic convolutional and transformer backbones; (ii) how LLM-guided subnet sampling and mutation affect training and search under matched evaluation budgets; and (iii) whether the resulting search adapts to heterogeneous hardware-specific accuracy–latency trade-offs.

5.1 EXPERIMENTAL SETUP

Datasets and Backbones. We evaluate the elastic backbones on CIFAR-10, CIFAR-100, and ImageNet-1K. The convolutional experiments use pretrained ResNet-based backbones (He et al., 2016), while the transformer experiments use ViT-B/16, DeiT-Small, and Swin-Tiny (Dosovitskiy et al., 2021; Touvron et al., 2021; Liu et al., 2021).

Search space. ELAX supports the following architecture spaces: the elastic-backbone space and the cell-based benchmark space. In the elastic-backbone space, each architecture variable selects a predefined block-level execution choice in a pretrained CNN or transformer. For hardware-benchmark evaluation, we use the NAS-Bench-201 cell space (Dong & Yang, 2020). It contains five operations on six directed edges, yielding $5^6 = 15,625$ architectures.

Hardware profiles. For the controlled hardware-aware experiments, we use the five HW-NAS-Bench (Li et al., 2021) profiles `edgegpu`, `raspi4`, `pixel3`, `eyeriss`, and `fpga`. Architecture accuracy is retrieved from NAS-Bench-201, while profile-specific latency is obtained from HW-NAS-Bench.

Training and search procedure. Each elastic supernet is trained for 100 epochs using stochastic gradient descent with momentum 0.9, an initial learning rate of 10^{-3}, weight decay 2×10^{-5}, batch size 32, cosine learning-rate scheduling, and label smoothing 0.1. We select the checkpoint achieving the highest validation accuracy according to the same criterion for all training variants. The evolutionary search uses a population size of 50 and a budget of 500 distinct architecture evaluations. LLM-guided mutation is applied with probability $\rho_{\mathrm{LLM}} = 0.2$, while the remaining offspring use standard evolutionary variation. We implement the hardware-aware LLM controller using Llama 3.1 8B (Dubey et al., 2024) through a local Ollama deployment.

Baselines. We compare ELAX with representative multi-objective evolutionary and Bayesian optimization methods for hardware-aware architecture search. SMS-EMOA (Beume et al., 2007) is a hypervolume-based evolutionary algorithm. qLogEHVI (Ament et al., 2023) is a batch multi-objective Bayesian optimization method that optimizes logarithmic expected hypervolume improvement. LaMOO+qLogEHVI combines the search-space partitioning strategy of LaMOO (Zhao et al., 2024) with qLogEHVI acquisition. MOBO-OSD-D is our discrete adaptation of MOBO-OSD (Ngo et al., 2025), which uses orthogonal search directions for batch multi-objective Bayesian optimization.

Evaluation metrics. We evaluate the hardware-specific Pareto set returned by each method using hypervolume (HV), inverted generational distance (IGD), and feasible Top-1 accuracy. These metrics are computed from the objectives $\mathbf{g}_h(\mathbf{a})$ defined in Eq. 3. For each profile h, we normalize each objective of $\mathbf{g}_h(\mathbf{a})$ to $[0, 1]$ using the minimum and maximum values over the NAS-Bench-201 search space for hardware profile h. Let $\widetilde{\mathbf{g}}_h(\mathbf{a})$ denote the normalized objective vector, and let $\mathcal{P}_h$ be the non-dominated architecture set returned by a method. The HV of $\mathcal{P}_h$ is

$$\mathrm{HV}(\mathcal{P}_h) = \lambda \left(\bigcup_{\mathbf{a} \in \mathcal{P}_h} [\widetilde{g}_{h,1}(\mathbf{a}), r_1] \times [\widetilde{g}_{h,2}(\mathbf{a}), r_2] \right), \tag{15}$$

where $\lambda(\cdot)$ denotes the area of the dominated region and $\mathbf{r} = (r_1, r_2) = (1.1, 1.1)$ is a reference point dominated by all normalized objective vectors. A larger HV indicates a better accuracy–latency Pareto front. We compute IGD between $\mathcal{P}_h$ and the exact Pareto front of NAS-Bench-201 under hardware profile h; lower IGD indicates a closer approximation to the optimal Pareto front. Feasible Top-1 accuracy is the highest validation accuracy among architectures in $\mathcal{P}_h$ that satisfy the predefined latency constraint for hardware profile h.

5.2 RESULTS

Hardware-Aware NAS results. Table 1 presents the hardware-aware NAS results on CIFAR-100. ELAX achieves the highest mean HV on `eyeriss` and `fpga`, and obtains near-best HV on `edgegpu` and `raspi4`. In particular, it reaches an HV of 1.17936 on `eyeriss`, compared with 1.17916 for LaMOO+qLogEHVI, and improves the strongest baseline mean on `fpga` from 1.16575 to 1.16669. These results demonstrate that ELAX effectively discovers high-quality accuracy–latency trade-offs across both mobile and specialized accelerator profiles.

Table 1: Hardware-aware NAS-Bench-201 architecture search results on CIFAR-100. All methods use the same budget of 500 unique architecture evaluations and hardware-specific latency measurements from HW-NAS-Bench. Feasible accuracy is the highest validation accuracy that satisfies the device-specific latency constraint, whereas best accuracy is the highest validation accuracy across all evaluated architectures without latency constraint. Values are mean ± standard deviation over ten runs with paired seeds across methods.

Profile	Method	HV ↑	IGD ↓	Feasible Acc. (%) ↑	Best Acc. (%) ↑
edgegpu	SMS-EMOA	1.09801 ± 0.00179	0.00978 ± 0.00177	71.523 ± 0.447	73.276 ± 0.215
	qLogEHVI	1.09818 ± 0.00153	0.01199 ± 0.00209	71.226 ± 0.362	73.320 ± 0.211
	LaMOO+qLogEHVI	**1.10090 ± 0.00045**	0.00695 ± 0.00158	71.804 ± 0.258	73.475 ± 0.059
	MOBO-OSD-D	1.09737 ± 0.00204	0.01211 ± 0.00146	71.434 ± 0.370	73.337 ± 0.220
	ELAX	1.10067 ± 0.00039	**0.00681 ± 0.00184**	**71.865 ± 0.100**	**73.493 ± 0.000**
eyeriss	SMS-EMOA	1.17772 ± 0.00146	0.01850 ± 0.01217	71.955 ± 0.084	73.217 ± 0.192
	qLogEHVI	1.17625 ± 0.00193	0.02196 ± 0.01143	71.904 ± 0.133	73.211 ± 0.240
	LaMOO+qLogEHVI	1.17916 ± 0.00067	0.00732 ± 0.00393	**72.019 ± 0.003**	73.381 ± 0.096
	MOBO-OSD-D	1.17676 ± 0.00174	0.01880 ± 0.00949	71.922 ± 0.092	73.233 ± 0.177
	ELAX	**1.17936 ± 0.00082**	**0.00429 ± 0.00376**	72.011 ± 0.030	**73.456 ± 0.079**
fpga	SMS-EMOA	1.16446 ± 0.00133	0.01927 ± 0.01244	70.702 ± 0.146	73.188 ± 0.300
	qLogEHVI	1.16399 ± 0.00138	0.01817 ± 0.00976	70.583 ± 0.212	73.272 ± 0.186
	LaMOO+qLogEHVI	1.16575 ± 0.00298	0.00951 ± 0.00921	70.766 ± 0.039	73.300 ± 0.435
	MOBO-OSD-D	1.16414 ± 0.00152	0.01463 ± 0.00486	70.521 ± 0.259	73.364 ± 0.151
	ELAX	**1.16669 ± 0.00071**	**0.00746 ± 0.00341**	**70.782 ± 0.025**	**73.456 ± 0.079**
pixel3	SMS-EMOA	1.15050 ± 0.00424	0.02267 ± 0.01608	70.900 ± 0.309	72.931 ± 0.412
	qLogEHVI	1.15020 ± 0.00264	0.01946 ± 0.00355	70.607 ± 0.301	73.178 ± 0.210
	LaMOO+qLogEHVI	**1.15842 ± 0.00159**	**0.00620 ± 0.00311**	**71.190 ± 0.032**	73.437 ± 0.090
	MOBO-OSD-D	1.15087 ± 0.00329	0.02046 ± 0.00421	70.598 ± 0.314	73.095 ± 0.213
	ELAX	1.15322 ± 0.00181	0.00687 ± 0.00349	71.091 ± 0.278	**73.456 ± 0.079**
raspi4	SMS-EMOA	1.14931 ± 0.00334	0.01371 ± 0.00447	70.900 ± 0.384	72.889 ± 0.461
	qLogEHVI	1.15105 ± 0.00207	0.01293 ± 0.00360	70.507 ± 0.564	73.321 ± 0.207
	LaMOO+qLogEHVI	**1.15545 ± 0.00057**	**0.00394 ± 0.00118**	71.147 ± 0.157	73.475 ± 0.059
	MOBO-OSD-D	1.14948 ± 0.00300	0.01296 ± 0.00499	70.675 ± 0.525	73.152 ± 0.311
	ELAX	1.15517 ± 0.00086	0.00418 ± 0.00123	**71.177 ± 0.136**	**73.493 ± 0.000**

ELAX also achieves the lowest mean IGD on edgegpu, eyeriss, and fpga, showing that its returned architecture sets closely approximate the exact device-specific Pareto fronts. On fpga, it reduces mean IGD from 0.00951 for LaMOO+qLogEHVI to 0.00746, a relative reduction of 21.6%. LaMOO+qLogEHVI obtains the strongest results on pixel3. Under the device-specific latency constraints, ELAX achieves the highest feasible validation Top-1 accuracy on edgegpu, fpga, and raspi4, while remaining within 0.10 percentage points of the best result on eyeriss and pixel3. Overall, the results show that ELAX adapts effectively to heterogeneous hardware targets and recovers competitive, high-quality Pareto sets under practical latency constraints.

Figure 3 shows the search behavior over the evaluation budget on FPGA. HV improves rapidly during the early evaluations and more gradually thereafter, with ELAX remaining close to the leading trajectories throughout the search. Across the five hardware profiles, ELAX obtains the largest mean area under the HV curve on edgegpu, eyeriss, and raspi4, while LaMOO+qLogEHVI leads on fpga and pixel3. The complete five-profile HV curves are provided in Appendix Figure 5, with the corresponding IGD and accuracy curves reported alongside them.

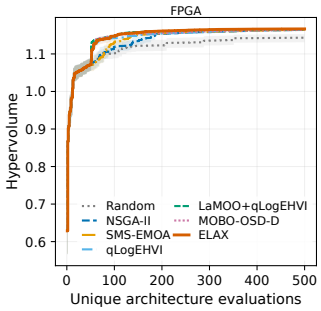
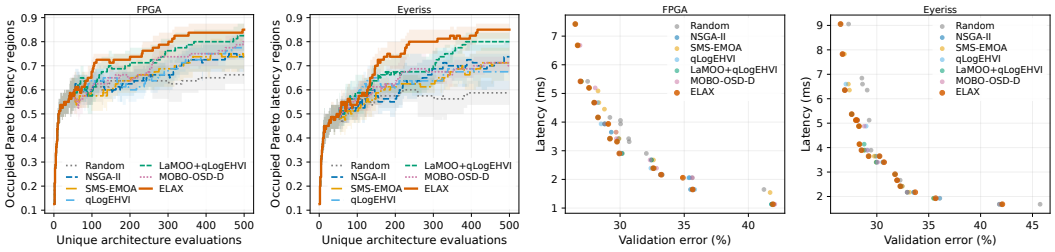


Figure 3: Anytime HV over NAS-Bench-201 architecture evaluations on FPGA.

Figure 4 further examines search coverage and the final Pareto sets on FPGA and Eyeriss. On both profiles, ELAX rapidly expands the number of occupied Pareto latency regions and maintains the broadest coverage over most of the evaluation budget. The resulting Pareto fronts contain competitive architectures across a wide range of latency budgets, providing multiple hardware-specific operating points. Complete five-profile coverage and Pareto-front results are provided in Appendix Figures 8 and 9.

Elastic transformer backbones. We next evaluate whether the proposed elastic-supernet construction and shared-weight training procedure generalize across pretrained transformer families.



(a) (b) (c) (d)

Figure 4: Search coverage and final Pareto fronts on FPGA and Eyeriss. (a)–(b) Occupied Pareto latency regions over the evaluation budget on FPGA and Eyeriss, respectively. (c)–(d) Final accuracy–latency Pareto fronts on FPGA and Eyeriss, respectively.

Table 3: ResNet-50/CIFAR-100 component study with $Q = 100$. Values are mean $\pm$ sample standard deviation over training seeds.

Variant	HV ↑	Selected Top-1 (%) ↑	Latency (ms) ↓
Full ELAX	0.74597 ± 0.00749	$\mathbf{80.20 \pm 0.82}$	4.197 ± 0.169
w/o LLM sampling	0.74867 ± 0.00474	78.56 ± 0.82	$\mathbf{4.142 \pm 0.094}$
w/o LLM-guided mutation	0.74715 ± 0.00536	79.13 ± 0.66	4.239 ± 0.131
w/o both	$\mathbf{0.74908 \pm 0.00208}$	79.49 ± 1.79	4.314 ± 0.090

Table 2 reports the full-path accuracy of ViT-B/16, DeiT-Small, and Swin-Tiny after elastic-supernet training. ViT-B/16 achieves the highest Top-1 accuracy on CIFAR-10 and CIFAR-100, whereas Swin-Tiny performs best on ImageNet-1K. These results verify that the proposed supernet construction and shared-weight training procedure can be applied to transformer backbones with different attention and hierarchical structures without being tied to a single architecture family. Additional ResNet results for elastic convolutional backbones are provided in Appendix A.5.

Table 2: Performance of ELAX with pretrained transformer backbones across different datasets.

Dataset	Backbone	Top-1 (%)	Top-5 (%)
CIFAR-10	ViT-B/16	**97.69**	99.96
	DeiT-Small	96.51	99.96
	Swin-Tiny	96.40	**99.97**
CIFAR-100	ViT-B/16	**87.82**	**98.59**
	DeiT-Small	81.30	97.45
	Swin-Tiny	80.68	97.11
ImageNet-1K	ViT-B/16	75.89	93.65
	DeiT-Small	78.38	94.41
	Swin-Tiny	**80.71**	**95.36**

LLM-guided sampling and mutation. We next examine the two uses of LLM guidance separately on ResNet-50/CIFAR-100. Table 3 examines the contributions of training-time LLM sampling and search-time LLM-guided mutation on ResNet-50/CIFAR-100. The full configuration achieves the highest mean selected validation accuracy. Removing LLM sampling or LLM-guided mutation reduces accuracy from 80.20% to 78.56% and 79.13%, respectively, while removing both gives 79.49%. The differences in HV are small, with the ablated variants obtaining slightly higher mean values. Thus, in this study, the two LLM components primarily improve the accuracy of the selected feasible subnet rather than the overall Pareto-front quality. Since one fixed search seed is used for each checkpoint, these results are descriptive; the ten-seed search comparison and complete hardware-ranking ablation are reported in Table 4 of Appendix A.4.

6 CONCLUSION

We presented ELAX, an LLM-guided evolutionary NAS framework that combines elastic-supernet training with hardware-aware multi-objective search. ELAX uses a shared LLM controller to guide subnet sampling during elastic-supernet training and propose local architecture edits during profile-specific evolutionary search, while deterministic validation and accuracy–latency evaluation govern candidate selection. Separating one-time shared-weight training from hardware-specific search enables the trained supernet to be reused across deployment profiles and each Pareto archive to support multiple latency budgets. Experiments on elastic transformer backbones and NAS benchmarks show that ELAX achieves competitive accuracy–latency trade-offs compared to baseline approaches across heterogeneous hardware profiles.

REFERENCES

E. Altman, J. Blanusa, et al. Are large language models good neural architecture generators for edge? In *IEEE International Conference on Edge Computing and Communications (EDGE)*, pp. 162–165, 2024.

Sebastian Ament, Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Unexpected improvements to expected improvement for bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Nicola Beume, Boris Naujoks, and Michael Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007. doi: 10.1016/j.ejor.2006.08.008.

Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

Han Cai Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

Z. Cai, L. Chen, P. Liu, T. Ling, and Y. Lai. EG-NAS: Neural architecture search with fast evolutionary exploration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 11159–11167, 2024.

Zhipeng Cai, Yihang Tang, Yuhong Lai, Hong Wang, Zhibo Chen, and Hong Chen. Seki: Self-evolution and knowledge inspiration based neural architecture search via large language models, 2025.

X. Chu, B. Zhang, R. Xu, and J. Li. FairNAS: Rethinking evaluation fairness of weight sharing neural architecture search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12239–12248, 2021.

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 (ICLR)*, 2020.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Abhimanyu Dubey et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

T. Gao, L. Guo, S. Zhao, P. Xu, Y. Yang, X. Liu, S. Wang, S. Zhu, and D. Zhou. QuantNAS: Quantization-aware neural architecture search for efficient deployment on mobile device. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 2210–2219, 2024.

Maxime Girard, Victor Quétu, Samuel Tardieu, Van-Tam Nguyen, and Enzo Tartaglione. Memory-optimized once-for-all network. In *Computer Vision – ECCV 2024 Workshops*, pp. 252–267, 2024.

Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Y. Hu, J. Liu, K. Wang, J. Zheng, W. Shi, M. Zhang, Q. Dou, R. Liu, A. Zhou, and H. Li. LM-Searcher: Cross-domain neural architecture search with LLMs via unified numerical encoding. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9408–9421, 2025.

J. Jeon et al. Subnet-aware dynamic supernet training for neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 30137–30146, 2025.

Zipeng Ji, Guanghui Zhu, Chunfeng Yuan, and Yihua Huang. RZ-NAS: Enhancing LLM-guided neural architecture search via reflective zero-cost strategy. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, pp. 27237–27254, 2025.

L. Jiang, S. D. Fu, Y. Zhu, and B. Li. Janus: Collaborative vision transformer under dynamic network environment. In *IEEE INFOCOM 2025 - IEEE Conference on Computer Communications*, pp. 1–10, 2025.

T. Jiang, H. Wang, and R. Bie. MeCo: Zero-shot NAS with one data and single forward pass via minimum eigenvalue of correlation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 76735–76747, 2023.

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

C. Li et al. Block-wisely supervised neural architecture search with knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1989–1998, 2020.

Chaojian Li, Zhongzhi Yu, Yonggan Fu, Yongan Zhang, Yang Zhao, Haoran You, Qixuan Yu, Yue Wang, Cong Hao, and Yingyan Lin. HW-NAS-Bench: Hardware-aware neural architecture search benchmark. In *International Conference on Learning Representations (ICLR)*, 2021.

J. Li, J. Zhang, F. Wang, and L. Ma. HEP-NAS: Towards efficient few-shot neural architecture search via hierarchical edge partitioning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 18314–18322, 2025.

Zhe Li, Zhiwei Lin, and Yongtao Wang. CoLLM-NAS: Collaborative large language models for efficient knowledge-guided neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 3273–3282, 2026.

M. Lin and J. Luo. Per-architecture training-free metric optimization for neural architecture search. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

J. Liu et al. Edge-cloud collaborative computing on distributed intelligence and model optimization: A survey. *IEEE Communications Surveys & Tutorials*, 2026.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021.

J. Lv et al. HyperNAS: Enhancing architecture representation for NAS predictor via hypernetwork. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12955–12965, 2026.

B. Lyu, H. Yuan, L. Lu, and Y. Zhang. Resource-constrained neural architecture search on edge devices. *IEEE Transactions on Network Science and Engineering*, 9(1):134–148, 2022.

Muhammad U. Nasir, Sam Earle, Julian Togelius, Steven James, and Christopher Cleghorn. LLMatic: Neural architecture search via large language models and quality-diversity optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 1110–1118, 2024.

U. Nath, Y. Wang, and Y. Yang. Neural architecture search finds robust models by knowledge distillation. In *Proceedings of the 40th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2024.

Lam Ngo, Huong Ha, Jeffrey Chan, and Hongyu Zhang. MOBO-OSD: Batch multi-objective bayesian optimization via orthogonal search directions. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Y. Peng, N. Du, H. Tang, T. Lu, and J. Long. Cream of the crop: Distilling prioritized paths for one-shot neural architecture search. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 17930–17940, 2020.

E. Real, P. Liang, et al. Evolutionary neural architecture search: A survey and recent advances. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 1–10, 2023.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4780–4789, 2019.

C. Tang et al. ElasticViT: Conflict-aware supernet training for deploying fast vision transformer on diverse mobile devices. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5829–5840, 2023.

P. K. Thind, V. Katturu, G. Zema, and R. Del Prete. Deployment-aligned low-precision neural architecture search for spaceborne edge ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10347–10357. PMLR, 2021.

Dilin Wang, Chengyue Li, Xianhiu Han, Min Cao, and Vikas Chandra. Alphanet: Improved training of supernets with alpha-divergence. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pp. 10760–10771, 2021.

H. Wang, B. Dedhia, and N. K. Jha. Zero-TPrune: Zero-shot token pruning through leveraging of the attention graph in pre-trained transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16070–16079, 2024.

Haoyuan Wang, Yang Gao, Xiaotian Zheng, Peng Zhang, Hong Chen, and Jiajun Bu. Graph neural architecture search with GPT-4, 2023.

Binhang Wu, Xiaoliang Dai, Peizhao Zhang, Yining Wang, Fei Sun, Yuxin Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10734–10743, 2019.

Y. Xue, C. Chen, and A. Słowik. Neural architecture search based on a multi-objective evolutionary algorithm with probability stack. *IEEE Transactions on Evolutionary Computation*, 27(4):778–786, 2023.

Y. Xue, Y. Lin, and F. Neri. Architecture knowledge distillation for evolutionary generative adversarial network. *International Journal of Neural Systems*, 2025.

Shan You You, Tao Huang, Mingmin Yang, Fei Wang, Chao Qian, and Chang Xu Zhang. Greedynas: Towards fast one-shot nas with greedy supernet. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

C. Yu et al. Progressive neural architecture generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 34524–34534, 2026.

L. Yu and W. Xiang. X-Pruner: Explainable pruning for vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24355–24363, 2023.

F. Zeng et al. An efficient hybrid vision transformer for TinyML applications. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19914–19924, 2025.

Yiyang Zhao, Linnan Wang, and Tian Guo. Multi-objective neural architecture search by learning search space partitions. *Journal of Machine Learning Research*, 25(177):1–41, 2024.

C. Zheng. iFormer: Integrating ConvNet and transformer for mobile application. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

Mingkai Zheng, Xiu Su, Shan You, Fei Wang, Chao Qian, Chang Xu, and Samuel Albanie. Can GPT-4 perform neural architecture search?, 2023.

H. Zhu, G. L. Zhang, and S. Huang. PEL-NAS: Search space partitioned architecture prompt co-evolutionary LLM-driven hardware-aware neural architecture search, 2025.

Barret Zoph and Quoc V. Le. Neural Architecture Search with Reinforcement Learning. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

A Appendix

A.1 Additional Experimental Results

This section supplements the end-of-budget results in the main paper with complete search trajectories, representative Pareto fronts, component analyses, and additional elastic-backbone results. In addition, the implementation and evaluation details are provided.

A.2 Search Trajectories and Pareto Fronts

The following figures complement the end-of-budget values in the main paper. They show the complete search trajectories, representative final Pareto fronts, component trajectories, and feasible-accuracy trajectories for benchmark spaces. The NAS-Bench-201 results are shown to preserve the legibility of the five-profile NAS-Bench-201 plots.

Figure 5 presents the hypervolume trajectories over the evaluation budget. Among the compared methods, ELAX obtains the highest mean HV area under the curve on `edgegpu`, `eyeriss`, and `raspi4`, whereas LaMOO+qLogEHVI is strongest on `fpga` and `pixel3`. Thus, the relative search efficiency varies across hardware profiles.

Figure 6 illustrates the corresponding IGD trajectories averaged over ten paired runs. ELAX obtains the lowest mean IGD area under the curve on all five hardware profiles, indicating that its evaluated Pareto sets remain close to the exact device-specific fronts over the search budget. The feasible-accuracy trajectories in Figure 7 further show the best validation accuracy found under each fixed device-specific latency constraint.

A.3 Final Pareto Sets and Latency-Range Coverage

Figure 8 visualizes representative final non-dominated architecture sets. For each method and hardware profile, the displayed set is obtained from the run with median final HV. The horizontal axis shows validation error, and the vertical axis shows HW-NAS-Bench lookup latency. Therefore, architectures closer to the lower-left corner offer more favorable accuracy–latency trade-offs.

The Pareto fronts differ substantially across hardware profiles. In particular, an architecture that is attractive under the `edgegpu` latency table may occupy a different operating region under the `raspi4`, `eyeriss`, or `fpga` latency tables. This observation motivates hardware-aware search rather than selecting a single architecture for every target device.

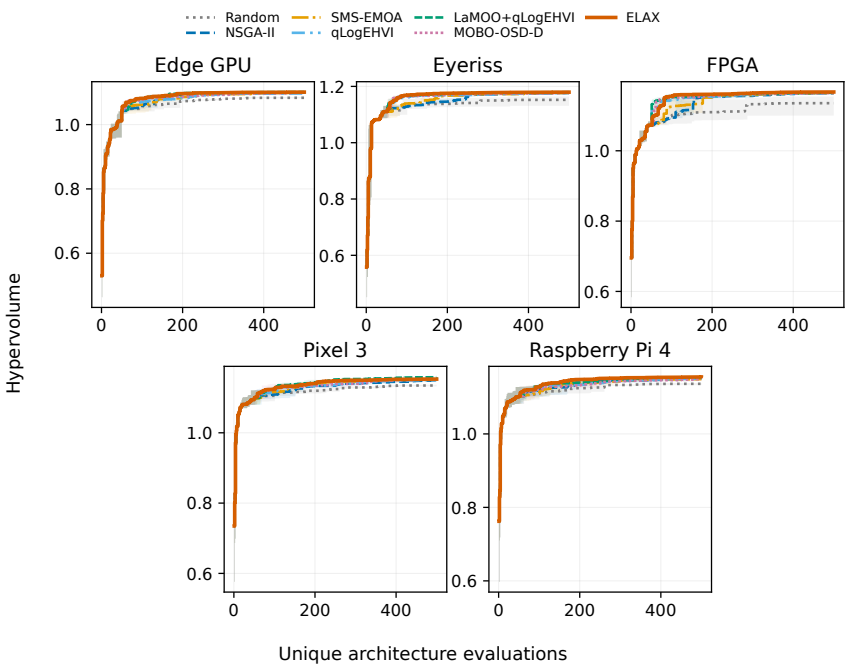


Figure 5: Hypervolume comparison of different methods on NAS-Bench-201/CIFAR-100 across five hardware profiles. The horizontal axis denotes the number of unique architecture evaluations. Curves show the mean over independent runs, and shaded regions indicate across-run uncertainty. Higher HV is better.

Figure 9 complements HV by the number of occupied latency regions in the current non-dominated archive. A larger value indicates that the returned Pareto set spans a broader range of feasible deployment options.

A.4 Effect of LLM-Based Parent–Mutation Ranking

Figure 10 compares ELAX with a matched random-ranking control. Both methods use the same parent pool, legal edit space, candidate-pair budget, and reserved evaluation budget. They differ only in whether the reserved parent–mutation candidates are ranked using LLM guidance or random ranking. The relative effect of LLM-based ranking varies across hardware profiles. ELAX achieves higher mean anytime HV on `edgegpu` and `eyeriss`, whereas the random-ranking control achieves higher mean anytime HV on `fpga`, `pixel3`, and `raspi4`. Thus, the effect of ranking depends on the hardware-specific search landscape and the available accuracy–latency regions.

Ablation study. Table 4 compares ELAX with a matched random-ranking control to evaluate the contribution of LLM guidance. At each guided mutation step, both variants generate the same number of legal parent–edit candidates from the same parent pool. ELAX uses the LLM to rank these candidates, whereas the control ranks them randomly. Both variants then apply the same surrogate screening and use the same evaluation budget, making the ranking strategy the only intended difference. Across ten paired runs, ELAX achieves lower mean IGD on `edgegpu`, `eyeriss`, `fpga`, and `raspi4`. Its effect on the other metrics varies across devices: ELAX obtains higher mean HV on `edgegpu` and `raspi4` and higher feasible accuracy on `edgegpu` and `fpga`, while the random-ranking control performs better on the remaining profiles. None of the paired differences is statistically significant after Holm correction. Overall, the results show that the effect of LLM ranking depends on the target hardware profile.

A.5 Elastic Convolutional and Transformer Backbones

Elastic convolutional and transformer subnets. We further examine the accuracy–latency tradeoffs exposed by the elastic backbones. For the convolutional setting, ELAX uses a pretrained ResNet-

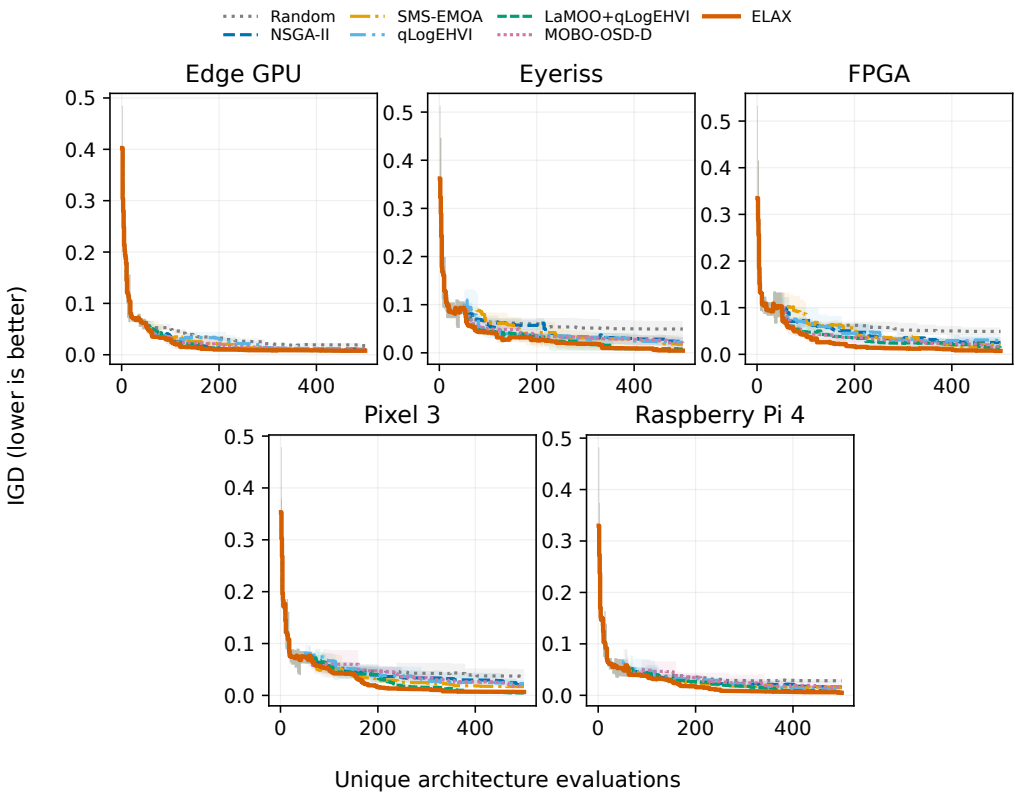


Figure 6: IGD Comparison results of ELAX with other approaches on NAS-Bench-201/CIFAR-100 across five hardware profiles. Lower IGD indicates a closer approximation to the reference Pareto front.

Table 4: Ablation of LLM-based parent–mutation ranking on hardware-aware NAS-Bench-201 search on CIFAR-100. The random-ranking control uses the same parent pool, legal edit space, pair count, and reserved evaluation quota.

Profile	Method	HV ↑	IGD ↓	Feasible Acc. (%) ↑
edgegpu	ELAX random-ranking	1.10043 ± 0.00088	0.00754 ± 0.00090	71.768 ± 0.300
	ELAX	$\mathbf{1.10067 \pm 0.00039}$	$\mathbf{0.00681 \pm 0.00184}$	$\mathbf{71.865 \pm 0.100}$
eyeriss	ELAX random-ranking	$\mathbf{1.17960 \pm 0.00056}$	0.00512 ± 0.00394	$\mathbf{72.020 \pm 0.000}$
	ELAX	1.17936 ± 0.00082	$\mathbf{0.00429 \pm 0.00376}$	72.011 ± 0.030
fpga	ELAX random-ranking	$\mathbf{1.16680 \pm 0.00040}$	0.00802 ± 0.00329	70.780 ± 0.033
	ELAX	1.16669 ± 0.00071	$\mathbf{0.00746 \pm 0.00341}$	$\mathbf{70.782 \pm 0.025}$
pixel3	ELAX random-ranking	$\mathbf{1.15379 \pm 0.00229}$	$\mathbf{0.00651 \pm 0.00297}$	$\mathbf{71.142 \pm 0.091}$
	ELAX	1.15322 ± 0.00181	0.00687 ± 0.00349	71.091 ± 0.278
raspi4	ELAX random-ranking	1.15497 ± 0.00091	0.00419 ± 0.00159	$\mathbf{71.208 \pm 0.038}$
	ELAX	$\mathbf{1.15517 \pm 0.00086}$	$\mathbf{0.00418 \pm 0.00123}$	71.177 ± 0.136

101 backbone with elastic execution choices distributed across residual blocks. Each ResNet-101 subnet is represented by a 33-dimensional binary or ternary architecture encoding. Figure 11 shows the discovered ResNet-101 subnet configurations on CIFAR-10 and CIFAR-100. On CIFAR-10, several lower-latency subnets retain more than 97% Top-1 accuracy. The CIFAR-100 frontier exhibits a clearer trade-off: higher-accuracy configurations generally require additional executed blocks and greater inference latency. Hence, the trained supernet provides multiple feasible operating points rather than a single fixed configuration. The corresponding transformer results in Fig. 12 exhibit the same general behavior. Across ViT-B/16, DeiT-Small, and Swin-Tiny, the searched subnets span different accuracy–latency operating points. The preferred subnet consequently depends on both the selected backbone and the available latency budget. These results demonstrate that the

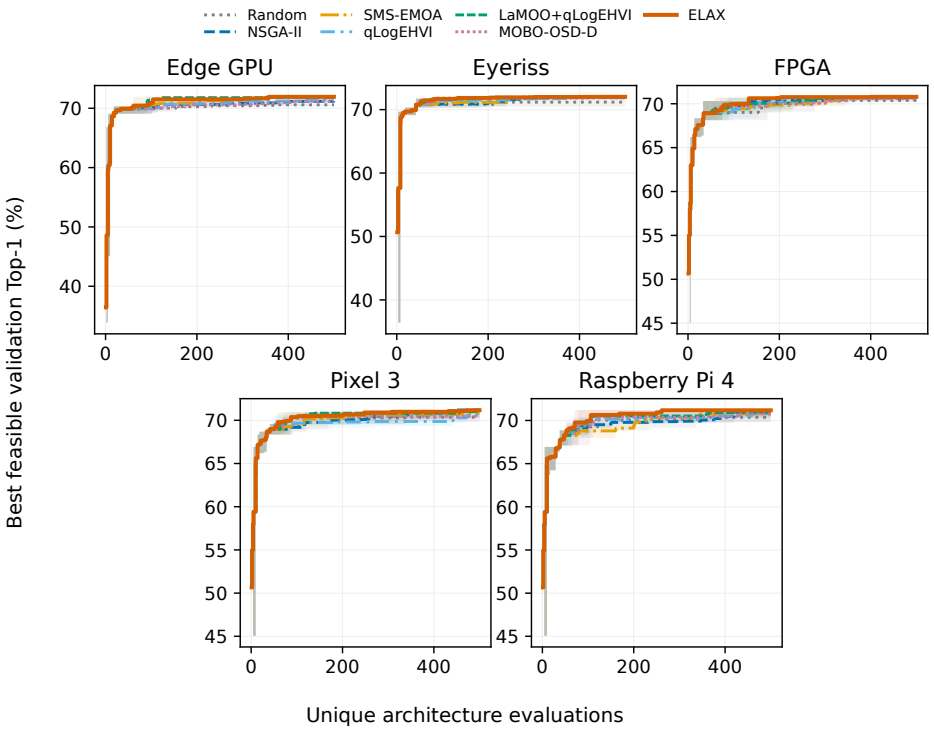


Figure 7: Best feasible validation accuracy found during the NAS-Bench-201/CIFAR-100 searches across five hardware profiles. The feasibility is determined by the fixed device-specific latency constraint.

Table 5: Validation and test accuracy of LLM-based NAS methods on NAS-Bench-201.

Method	CIFAR-10		CIFAR-100		ImageNet16-120	
	Valid	Test	Valid	Test	Valid	Test
GENIUS Zheng et al. (2023)	91.07 ± 0.20	93.79 ± 0.09	70.96 ± 0.33	70.91 ± 0.72	45.29 ± 0.81	44.96 ± 1.02
LLMatic Nasir et al. (2024)	91.42 ± 0.13	94.26 ± 0.10	71.41 ± 1.44	71.62 ± 1.73	44.98 ± 0.87	45.87 ± 0.96
RZ-NAS Ji et al. (2025)	91.45 ± 0.10	94.24 ± 0.12	73.35 ± 0.14	73.30 ± 0.21	46.53 ± 0.24	46.24 ± 0.23
SEKI Cai et al. (2025)	91.44 ± 1.26	94.34 ± 0.35	72.74 ± 1.92	72.75 ± 0.37	46.56 ± 0.56	$\mathbf{46.90 \pm 0.19}$
LM-Searcher Hu et al. (2025)	91.52	94.20	72.82	72.96	46.48	46.51
CoLLM-NAS Li et al. (2026)	$\mathbf{91.59 \pm 0.04}$	$\mathbf{94.37 \pm 0.01}$	$\mathbf{73.44 \pm 0.12}$	$\mathbf{73.44 \pm 0.15}$	$\mathbf{46.62 \pm 0.10}$	46.79 ± 0.28
ELAX (ours, $Q = 100$)	$\mathbf{91.59 \pm 0.03}$	$\mathbf{94.37 \pm 0.01}$	73.34 ± 0.27	73.30 ± 0.38	46.61 ± 0.10	46.46 ± 0.22
optimal	91.61	94.37	73.49	73.51	46.77	47.31

elastic training and search interfaces can be instantiated for convolutional and transformer architecture spaces.

A.6 Contextual Comparison with LLM-Based NAS Methods

LLM-based NAS methods comparison on NAS-Bench-201. As shown in Table 5, ELAX achieves strong and consistent performance across all three NAS-Bench-201 datasets under a query budget of $Q = 100$. On CIFAR-10, ELAX matches CoLLM-NAS in both validation and test accuracy, attaining 91.59% validation accuracy and 94.37% test accuracy, which also matches the NAS-Bench-201 test optimum. Compared with GENIUS and LLMatic, ELAX achieves higher accuracy on all six validation and test metrics. It also outperforms RZ-NAS on four metrics, matches its CIFAR-100 test accuracy, and differs by only 0.01 percentage points on CIFAR-100 validation accuracy. Furthermore, ELAX exceeds LM-Searcher on five of the six reported metrics. Although CoLLM-NAS remains slightly higher on CIFAR-100 and ImageNet16-120, the margins are small, demonstrating that the proposed approach is highly competitive with recent LLM-based NAS approaches.

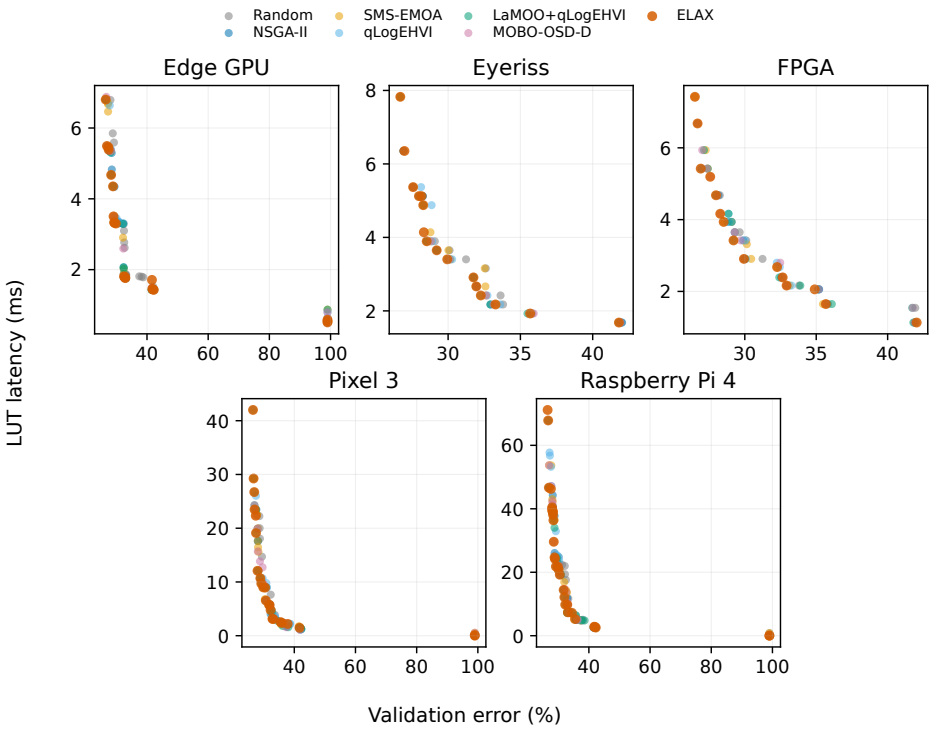


Figure 8: Final accuracy–latency Pareto fronts for NAS-Bench-201/CIFAR-100. Each panel corresponds to one hardware profile. For each method, the shown set is taken from the median final-HV run. The axes show validation error and HW-NAS-Bench lookup latency.

A.7 Reproducibility Details

Benchmark and metric protocol. We use NAS-Bench-201 on CIFAR-100 and HW-NAS-Bench latency lookup tables for `edgegpu`, `eyeriss`, `fpga`, `pixel3`, and `raspi4`. NAS-Bench-201 contains 15,625 discrete cell architectures; we enumerate the complete search space to construct the exact hardware-specific reference Pareto front. Architecture accuracy is taken from the NAS-Bench-201 validation record, and latency is read from the corresponding hardware lookup table. Both objectives are normalized to $[0, 1]$ using their minimum and maximum values over the complete search space. HV uses the reference point $(1.1, 1.1)$. IGD is computed with respect to the exact hardware-specific Pareto front obtained by enumerating all 15,625 architectures.

The latency constraint is fixed before search and is independent of architecture accuracy. Specifically, we use the median latency of the complete device-specific lookup table: 5.329 ms for `edgegpu`, 5.123 ms for `eyeriss`, 3.725 ms for `fpga`, 11.756 ms for `pixel3`, and 25.481 ms for `raspi4`. Duplicate architectures are rejected globally and do not consume the evaluation budget. Table 6 summarizes the common search and LLM-controller settings used throughout the hardware-aware NAS experiments.

A.8 LLM Controller Specification

The LLM does not generate architectures or operators directly. Instead, it ranks a finite set of legal parent stage template edits constructed by the search algorithm. This design ensures that every accepted LLM proposal maps to an executable architecture in the predefined NAS-Bench-201 search space.

At each controller update, the search selects up to eight region-relevant parent architectures from the current archive. Each parent is represented by its architecture encoding, validation accuracy, normalized error, latency, normalized latency, and archive-membership indicator. The prompt additionally contains the target Pareto region, the legal template menu, valid stage indices, and evidence derived

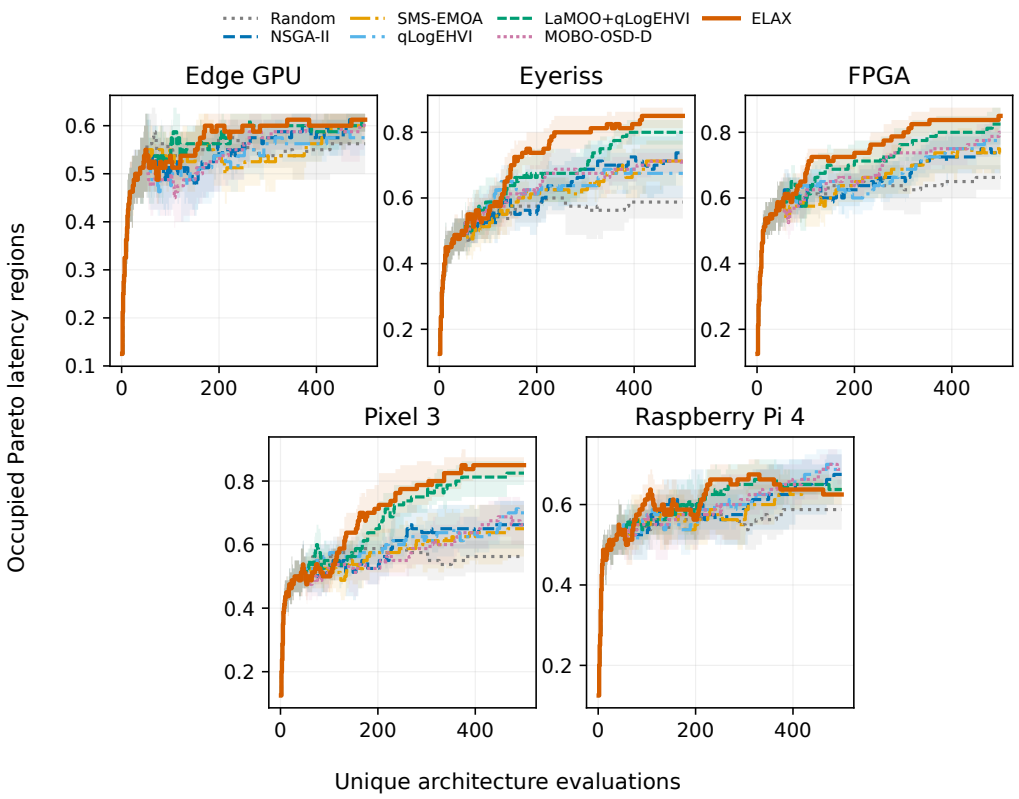


Figure 9: Coverage of Pareto latency regions. The non-dominated archive is partitioned into eight normalized latency regions; larger values indicate that the discovered Pareto set covers a broader range of accuracy–latency operating points.

Table 6: Implementation configuration for hardware-aware NAS on NAS-Bench-201/CIFAR-100.

Component	**Setting**
Search space	NAS-Bench-201, 15,625 architectures
Hardware measurements	HW-NAS-Bench lookup latency
Hardware profiles	`edgegpu`, `eyeriss`, `fpga`, `pixel3`, and `raspi4`
Population size	50
Evaluation budget	500 unique architectures
Crossover probability	0.9
Feedback batch size	5 architectures
Candidate-pool size	256
Rank model	Pairwise Random Forest ranker
qLogEHVI Monte Carlo samples	64
Prompt parent count	8 non-dominated or region-relevant parents
Requested LLM edits	12 distinct legal edits
LLM-reserved evaluations	1 of every 5 evaluated proposals
LLM backend and model	Ollama with llama3.1:8b
Request timeout	180 s

solely from previously evaluated architectures. The evidence window contains the most recent 64 relevant search observations. No reference-front information, unevaluated architecture accuracy, or unregistered operators is exposed to the LLM.

For the NAS-Bench-201 hardware-aware experiments, no supernet-derived evidence is provided to the controller. The LLM conditions only on benchmark observations accumulated during the current search. A region-specific edit plan is refreshed when that region is revisited after at least 50

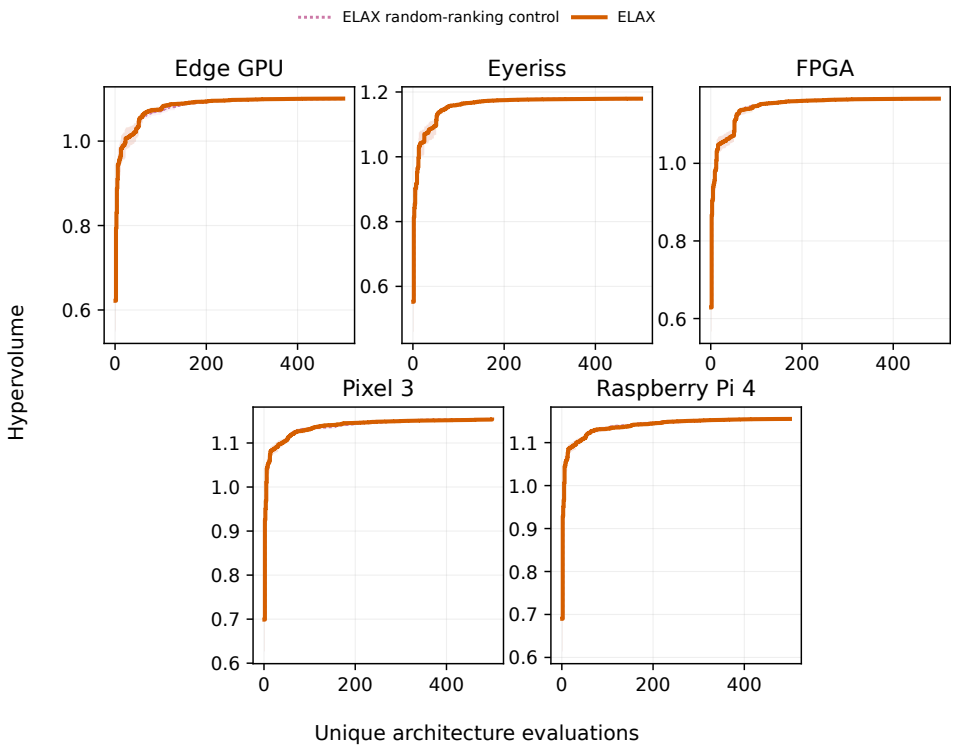
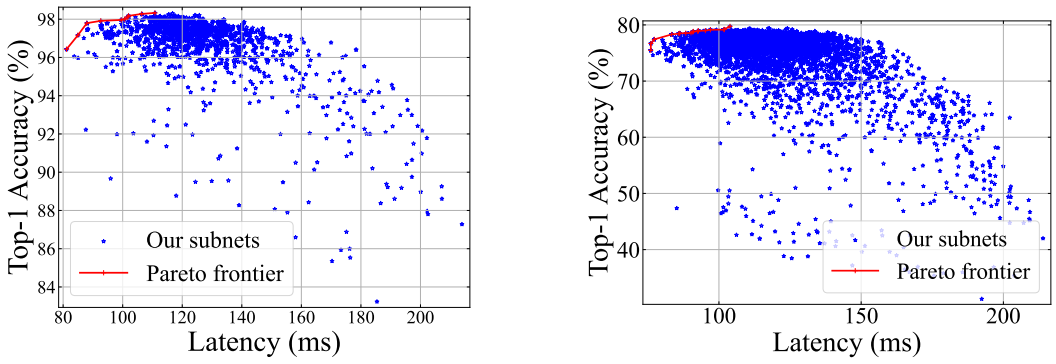


Figure 10: Hypervolume trajectories for the ELAX component controls on NAS-Bench-201/CIFAR-100 across five hardware profiles.



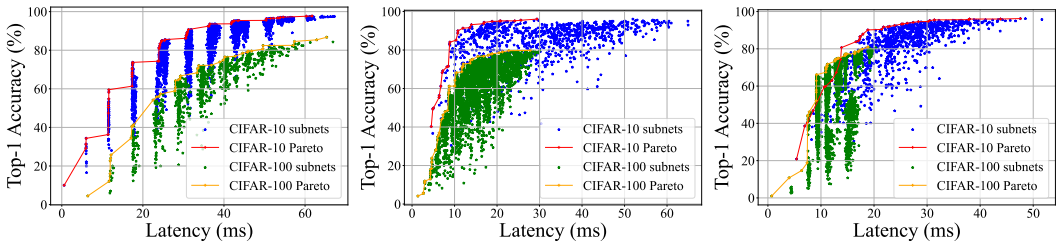
(a) On CIFAR-10. (b) On CIFAR-100.

Figure 11: Accuracy–latency Pareto frontiers identified from the elastic ResNet-101 backbone on CIFAR-10 and CIFAR-100.

additional evaluations. The controller requests 12 distinct legal edits and reserves one of every five evaluated proposals for the LLM-guided source; this candidate is subsequently screened using the same qLogEHVI-based acquisition procedure as the remaining candidates.

Figure 13 presents the static instruction used in every LLM request. Fields enclosed in brackets are instantiated from the current search state.

Output validation and fallback. A response is accepted only when semantic validation confirms the required number of distinct legal edits. Each `parent_index` must refer to a parent supplied in the request; each `stage` must lie in the permitted range; each `template_id` must occur in



(a) ViT-B/16. (b) Swin-Tiny. (c) DeiT-Small.

Figure 12: Accuracy–latency Pareto frontiers identified by ELAX for three transformer backbones on CIFAR-10 and CIFAR-100: (a) ViT-B/16, (b) Swin-Tiny, and (c) DeiT-Small.

Role. You are the evidence-guided mutation ranker in a hardware-aware multi-objective neural architecture search algorithm. Rank only legal parent stage template edits. Never invent an operation, parent, stage, or template. The downstream evaluator validates every edit and measures its true objectives.

Objective. Improve the accuracy–latency Pareto front by prioritizing edits that are likely to increase validation accuracy without losing hardware feasibility. Hypervolume rewards both higher accuracy and lower latency.

Search context. Benchmark: `[benchmark]`; dataset: `[dataset]`; hardware profile: `[device]`; target Pareto region: `[region]`;
Previously evaluated search evidence: `[search_evidence]`;
Supernet-derived evidence, when available: `[phase1_evidence]`;
Candidate parents: `[parents]`;
Legal template menu: `[menu]`;
Valid stage indices: 0 through `[last_stage]`.

Decision rules.
1. Prefer parent–stage–template combinations supported by positive empirical accuracy change, positive exclusive hypervolume, Pareto-archive entry, and the target latency direction.
2. Avoid selecting extremely low-accuracy parents solely because they are archive members or have low latency when more accurate legal parents are available.
3. For capacity-oriented regions, favor plausible accuracy gains while allowing a modest latency increase. For efficiency-oriented regions, favor latency reduction only when accuracy is likely to be preserved.
4. When evidence is sparse, use the parent's measured accuracy and latency together with the operation semantics; do not assume that a capacity-increasing edit is always beneficial.
5. Preserve diversity across parents, stages, and template families without sacrificing strong empirical evidence for Pareto improvement.
6. Assign larger weights to edits with stronger evidence of improving the accuracy–latency Pareto front.

Output requirement. Return one complete JSON object containing an `edits` array of `[minimum_count]` to `[maximum_count]` distinct edits. Each edit must contain `parent_index`, `stage`, `template_id`, and `weight`. Copy `template_id` verbatim from the legal menu. The weight must be finite and lie in $(0, 1]$. Do not include objective predictions, reasoning, or explanations.

Figure 13: Prompt used by the LLM controller. Bracketed fields are instantiated from the current search state. The controller ranks only legal parent stage template combinations and cannot introduce unsupported operators or architectures.

the legal menu; and each weight must be finite and lie in $(0, 1]$. Accepted edits are ordered by descending weight, with response order used to break ties.

If the parsed response contains fewer than 12 legal distinct edits, the controller issues a corrective request that restates the allowed parent indices, stage range, and template identifiers. At most two corrective requests are made. If a valid response is not obtained, the reserved proposal is completed using the predefined non-LLM candidate pool, and the event is recorded. The protocol limits fallback rates to 10% for LLM requests, 5% for acquisition, and 10% for the LLM-reserved proposal slot.

A.9 AI USE DISCLOSURE

We used a large language model (LLM) to polish the writing and provide editorial assistance while preparing this manuscript. This assistance included improving the grammar, readability, and conciseness of text written by the author. All AI-assisted suggestions were carefully reviewed, revised where necessary, and verified by the author before inclusion in the manuscript.