LLMFORGE: MULTI-BACKEND HARDWARE-AWARE NEURAL ARCHITECTURE SEARCH WITH INFINITE-HEAD ATTENTION FOR EDGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Efficient language model deployment requires architectures that balance model quality with the energy and latency costs of the target hardware. We present LLMForge, a hardware-aware neural architecture search framework for pretrained language models. Its Infinite-Head Attention parameterization decouples query-head count from head dimensions and allows separate query/key and value widths. Together with variable key–value grouping and MLP widths, this defines a flexible search space for capacity allocation within and across layers. To evaluate candidates efficiently, LLMForge converts each pretrained model into an elastic supernet and jointly trains sampled subnetworks with shared weights, avoiding separate candidate training during search. Four interchangeable backends provide GPU measurements, accelerator modeling, joint model–chip optimization, and cost prediction from smartwatch measurements. Their cost evaluations guide multi-objective search alongside supernet validation loss, allowing the selected architectures to adapt to both the hardware backend and the optimization objectives. We evaluate the framework using five pretrained models spanning 135M–4B parameters. Across eight model–accelerator configurations, the search reduces modeled energy per token by a median of 8.4% relative to uniform scaling at matched supernet validation loss.

1 INTRODUCTION

Deploying transformer language models (Vaswani et al., 2017) on edge devices and cost-constrained servers requires balancing model quality against limited computational and memory resources. The relevant constraints depend on the deployment setting: mobile devices operate within tight energy and thermal budgets (Laskaridis et al., 2024), while server workloads must balance inference cost with response time and throughput. These requirements make architectural design a deployment-specific problem: the capacity allocated to different model components must support language modeling quality while making effective use of the target hardware.

Exploring this allocation requires a sufficiently flexible architecture search space. Common attention configurations, however, couple head count and head dimension through a fixed total attention width and assign the same dimension to query, key, and value heads. Grouped-query attention (Ainslie et al., 2023) adds flexibility by allowing query heads to share key–value groups, but does not by itself remove these dimensional couplings. Search spaces built on these configurations therefore exclude potentially useful allocations of attention capacity, even when they satisfy the same deployment budget.

Allowing these architectural choices to vary across layers creates a combinatorial search space in which training every candidate separately is impractical. Shared-weight supernets reduce this cost and can be constructed from pretrained language models (Cai et al., 2020; Wang et al., 2020; Cai et al., 2024). Extending this approach to the broader space requires supporting changes in head dimensions and key–value group counts within the same set of shared parameters.

Alongside candidate quality, the search needs to evaluate execution costs on the target hardware. These costs depend on memory traffic, hardware utilization, and runtime implementation, whose effects vary across platforms and are not fully captured by parameter count or FLOPs. Cost evaluation

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.

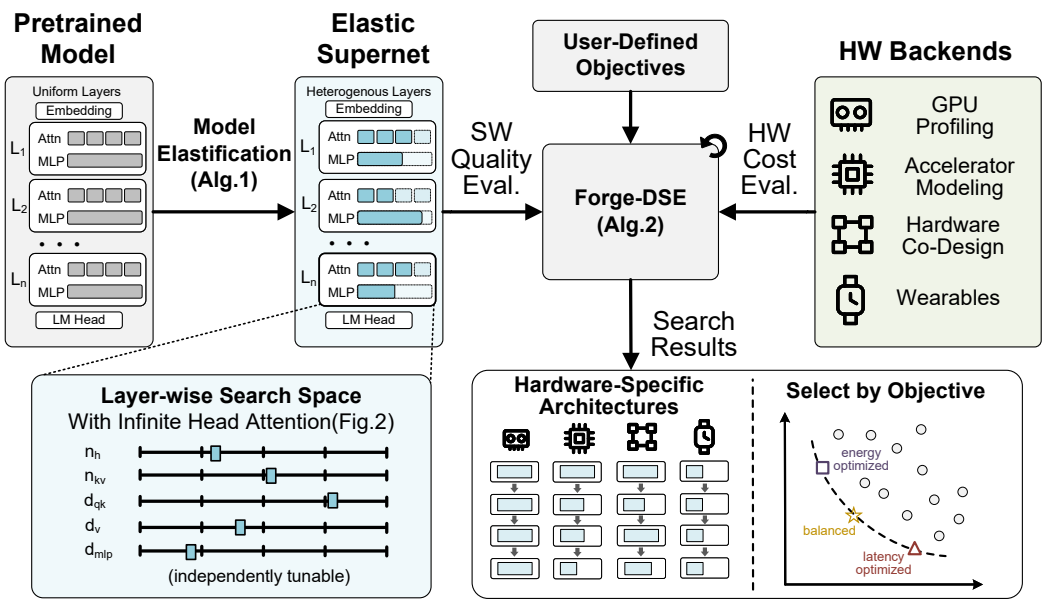


Figure 1: Overview of LLMForge. A pretrained model is elastified once to support shared-weight evaluation of IHA candidates. During search, the selected hardware backend supplies deployment costs, and the specified objectives and constraints guide architecture selection.

therefore needs to account for the execution characteristics of each target, with energy and latency metrics chosen according to the deployment objectives and workload.

We present **LLMForge**, a hardware-aware architecture search framework for adapting pretrained language models to deployment objectives and constraints (Figure 1). Its search engine, **Forge-DSE**, returns Pareto fronts that expose trade-offs between estimated model quality and hardware cost. Our contributions are as follows.

- **Infinite-Head Attention (IHA).** We decouple attention head count from head dimensions and allow separate query/key and value widths. Together with variable key–value grouping and MLP width, this defines a layer-wise search space.
- **Model elastification.** We convert five pretrained models spanning 135M–4B parameters into elastic supernets through coordinated weight slicing, learned group alignment, and a single shared-weight training run per model. Their subnetworks provide validation-loss estimates without separate candidate training.
- **Forge-DSE multi-backend search engine.** We develop Forge-DSE, an NSGA-II-based multi-objective search engine that uses four interchangeable hardware backends to select architectures for different deployment targets and objectives. Across eight model–accelerator configurations, it reduces modeled energy per token by a median of 8.4% relative to uniform scaling at matched supernet validation loss.

2 METHOD

We describe IHA and the search space in §2.1, elastification in §2.2, the backends in §2.3 and the search engine in §2.4.

2.1 INFINITE-HEAD ATTENTION AND THE SEARCH SPACE

IHA parameterizes attention by $(n_h, n_{kv}, d_{qk}, d_v)$, denoting the query head count, key–value group count, query/key head dimension, and value head dimension, respectively. As illustrated in Figure 2, n_h, d_{qk}, and d_v can vary independently without tying the attention widths to d_{model}, while $n_{kv} \mid n_h$ ensures that each key–value group serves an equal number of query heads. The group count and head

dimensions also determine KV-cache storage, which requires $n_{kv}(d_{qk} + d_v)$ elements per token per layer.

We combine these attention parameters with the MLP width d_{mlp} to define the search space for each transformer layer, restricting all choices to configurations supported by the elastic supernet (§2.2). Within this space, n_{kv} is selected from the divisors of the base model's key–value group count, and n_h from compatible head counts in the search grid. The query/key, value, and MLP widths each take $\{1/4, 1/2, 3/4, 1\}$ of their respective base widths, allowing capacity to vary across layers while preserving model depth and residual width. Appendix B specifies the grids for each base model.

2.2 Model Elastification

We convert a pretrained language model into an elastic supernet and jointly train sampled subnetworks with shared weights. These weights then support candidate evaluation without training each architecture separately.

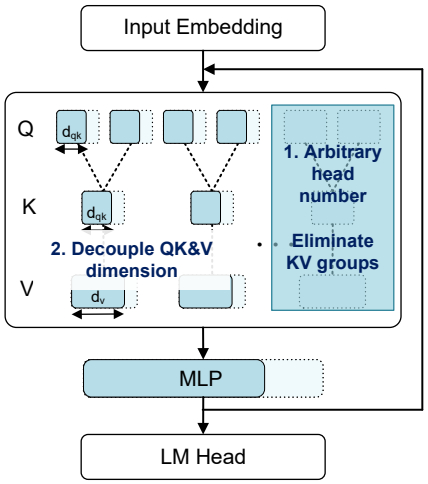


Figure 2: Infinite-Head Attention (IHA) Architecture.

Coordinated slicing. Each subnetwork selects query heads evenly across the active key–value groups, with corresponding selections in the output projection. Query/key dimensions are gathered as complete RoPE (Su et al., 2024) coordinate pairs in a fixed order alternating between high and low frequencies. The same indices select query/key channels, normalization gains, and RoPE frequencies, keeping these operations consistent as head dimensions change. QK-Norm is computed over the retained dimensions, and each layer learns an attention-scaling parameter for every allowed d_{qk}, initialized to preserve the standard scaling. Value channels and MLP hidden units use nested prefixes, with matching selections in their associated projections.

Group alignment and pooling. To reduce n_{kv}, we merge contiguous bundles of pretrained key–value groups by mean pooling (Ainslie et al., 2023). Before pooling, learnable rotations align their representations through independent two-dimensional rotations within each head. Query/key rotations follow the selected RoPE coordinate pairs, whereas value rotations pair coordinates $(p, p + d_v/2)$ and therefore use separate angles for each value width. The angles are also indexed by the active group count and original group, while projection weights remain shared across configurations.

Each retained query head uses the key-side rotation of its source group, identified by the active head-selection map. Query/key rotations follow QK-Norm, and RoPE is applied after the aligned keys and values have been pooled. After attention, each head's output undergoes the inverse value-side rotation of the same source group before the output projection. These transformations construct the current subnetwork without permanently merging shared groups; pooling and rotations are bypassed when all original groups are retained.

Supernet training. Algorithm 1 combines sandwich sampling with distillation from the frozen checkpoint M. For each micro-batch, SampleSandwich returns the full configuration, the minimum configuration, and $K = 2$ random configurations, indexed by $c = 0, \ldots, K + 1$. Writing $p_\theta^\tau(x; a_c)$ for the subnetwork's token distributions at temperature τ and $q = p_M^\tau(x)$ for the teacher's predictions, we use

$$\mathcal{L}_c(x, y; q) = \mathrm{CE}\big(y, p_\theta^1(x; a_c)\big) + \lambda_c \tau^2 D_{\mathrm{KL}}(q \,\|\, p_\theta^\tau(x; a_c)) \,, \tag{1}$$

where $\lambda_0 = 0$, $\lambda_c = \lambda_{\mathrm{KD}}$ for $c > 0$, and $\tau = 2$. Thus, the full pass learns directly from token labels, while the remaining passes also receive teacher supervision. Both loss terms are averaged over valid tokens.

Random configurations share one shape across layers during the first 10% of optimizer steps and subsequently vary independently by layer. Gradients are averaged across configurations and A

micro-batches, clipped to a global norm of 1.0, and applied in one optimizer update. Appendix C specifies the sampling distributions, optimizer settings, and document-aware training protocol.

Algorithm 1 Elastic supernet training

Require: Frozen checkpoint M, search space $\mathcal{S}$, steps N, accumulation count A, KD weight λ_{KD}
Ensure: Trained supernet parameters θ

Initialize θ from M, with identity rotations and standard attention scaling
for $t = 0, \ldots, N-1$ **do**
  $g \leftarrow 0$
  for $b = 1, \ldots, A$ **do**
    $(x, y) \leftarrow \textsc{NextMicroBatch}$; $q \leftarrow p_M^{\tau}(x)$
    $\mathcal{C} \leftarrow \textsc{SampleSandwich}(\mathcal{S}, K, t/N)$
    for $c = 0, \ldots, K+1$ **do**
      $a_c \leftarrow \mathcal{C}[c]$
      $g \leftarrow g + \dfrac{\nabla_\theta \mathcal{L}_c(x, y; q)}{A(K+2)}$
    end for
  end for
  $g \leftarrow g / \max\{1, \|g\|_2\}$ ▷ Clip after accumulation
  $\theta \leftarrow \textsc{AdamWUpdate}(\theta, g, t)$
end for
return θ

Candidate evaluation. The search scores each inherited subnetwork by mean token cross-entropy on 32 held-out FineWeb-Edu documents (Penedo et al., 2024), each truncated to 1024 tokens, taking 1.2–1.9 seconds per candidate on one GPU. To reduce selection bias, the final Pareto fronts are re-scored on a disjoint set of 64 held-out documents.

2.3 Hardware Backends

The backends share the interface $\mathrm{HW}(x, \mathcal{W}) \rightarrow (E_{\mathrm{tok}}, \mathrm{TTFT}, \mathrm{TPOT})$, which evaluates architecture x under a prefill–decode workload $\mathcal{W}$ and returns energy per token, time to first token, and time per output token.

GPU Profiling (Backend A). We measure candidate energy and latency through execution on an NVIDIA GPU, using ZEUS (You et al., 2023) for energy measurement. Appendix D details the measurement protocol.

Accelerator Modeling (Backend B). As shown in Figure 3, a workload analyzer decomposes each candidate into primitive operators. Timeloop (Parashar et al., 2019) maps these operators onto a target accelerator specification, and their costs are aggregated into workload-level energy and latency. We model Gemmini-, Eyeriss-, and FLAT-style accelerators (Genc et al., 2021; Chen et al., 2019; Kao et al., 2023), along with the single-chip decoder execution engine (DXE) (Tao et al., 2026) used by Backend C (Appendix E).

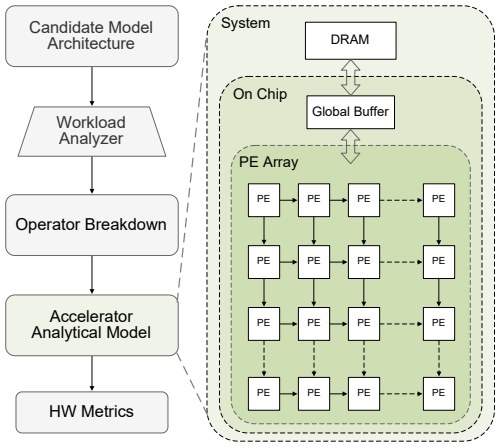


Figure 3: Operator-level workload analysis and accelerator cost modeling in Backend B.

Hardware Co-Design (Backend C). This backend models a ring-based DXE (rDXE) that pipelines tokens across chips with stationary weights. Per-layer weight memory, KV-cache requirements, and compute demand guide balanced packing of contiguous layers onto ring stages for each feasible chip configuration (Figure 4). The simulator combines single-chip execution with ring scheduling to estimate costs for joint architecture and chip search (Appendix F).

Smartwatch (Backend D). We measured 2,000 per-layer configurations of SmolLM2-135M and SmolLM2-360M on a Pixel Watch 5 with an INT8 C inference runtime, and fit a cost predictor, one XGBoost regressor per cost in log space on 110 architecture-only features. On 200 held-out architectures, the predictor reaches a mean absolute percentage error of 14.0% on energy per token, 16.5% on TTFT and 15.5% on TPOT, with rank correlations of 0.94, 0.92 and 0.84. Since the search compares candidates rather than reading absolute costs, ranking matters most. The predictor orders 89.6%, 88.9% and 83.3% of held-out pairs correctly on the three costs. Appendix G gives the measurement workload, the split and the training details.

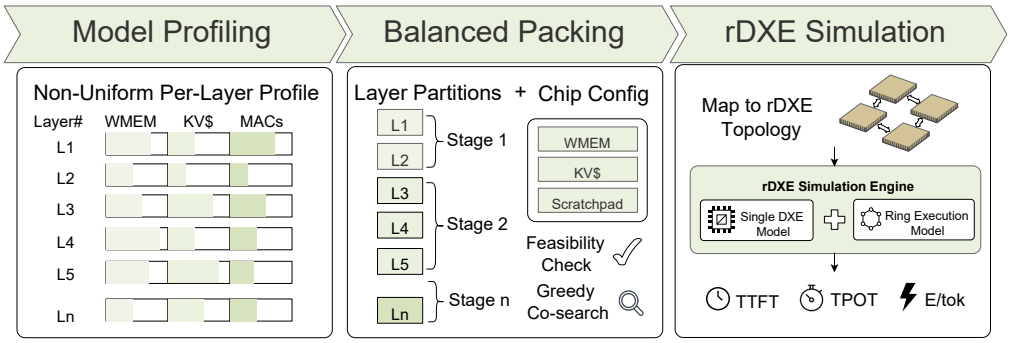


Figure 4: Joint architecture and chip evaluation through layer profiling, balanced packing, and ring simulation in Backend C.

2.4 Multi-Objective Search

Forge-DSE optimizes architectures under user-specified objectives and constraints. To reduce the search dimensionality, attention configurations and MLP widths are shared within separately defined blocks of contiguous layers, preserving capacity variation across the network (Appendix B).

At each generation, NSGA-II (Deb et al., 2002) produces offspring through crossover and mutation, then selects the next population from parents and offspring using constrained non-dominated sorting (Algorithm 2). Feasible candidates take priority over infeasible ones, which are ranked by constraint violation, and within each front, crowding distance favors candidates in less crowded regions of the objective space to maintain diverse trade-offs.

The EvaluateNew routine evaluates uncached candidates using supernet validation loss and backend hardware metrics, storing their objective values and constraint violations in an archive. Cached evaluations can be reused across searches when the supernet, validation data, and hardware evaluation settings remain unchanged. Search hyperparameters are provided in Appendix H.

3 Experiments

We evaluate LLMForge using five pretrained base models spanning 135M to 4B parameters: SmolLM2-135M and SmolLM2-360M (Allal et al., 2025), and Qwen3-0.6B, Qwen3-1.7B, and Qwen3-4B (Yang et al., 2025). Appendix A summarizes the model–backend combinations, inference workloads, and search objectives, and Appendix B specifies the search grid for each base model.

To assess the benefit of architecture search, we first compare searched architectures with uniformly scaled baselines at matched validation loss (§3.1). We then examine how hardware characteristics affect architecture selection (§3.2) and whether the deployment objectives a search is given change that selection (§3.3).

3.1 Search Results Across Backends and Scales

Figure 5 illustrates search progress across the four backend classes on the five-knob elastic-KV search space. The ablation below isolates what that space contributes.

Algorithm 2 Forge-DSE multi-objective search.

Require: Search space $\mathcal{S}$, supernet θ, backend HW, workload $\mathcal{W}$, objectives O, constraints C
Require: Population size N, offspring count M, generations G, retry limit τ, variation probabilities p_c, p_m, p_s
Ensure: Feasible non-dominated architectures after final rescoring

$\mathcal{A} \leftarrow \emptyset$; $P \leftarrow$ {full architecture, smallest architecture}
Fill P to size N with unique uniform samples from $\mathcal{S}$
EVALUATENEW($P, \mathcal{A}$)
for $g = 1, \ldots, G$ **do**
    $Q \leftarrow [\,]$
    for $i = 1, \ldots, M$ **do**
        $x \leftarrow$ VARY(P), with at most τ retries for duplicates in $\mathcal{A}$ or Q
        Append x to Q
    end for
    EVALUATENEW($Q, \mathcal{A}$)
    $P \leftarrow \text{NSGAIISelect}(\text{Unique}(P \cup Q), N; O, C)$
end for
$F \leftarrow \text{Rescore}(\text{FeasiblePareto}(\mathcal{A}))$ ▷ independent held-out data
return the non-dominated architectures in F

procedure VARY(P)
    Select parents a, b by two constrained binary tournaments
    $x \leftarrow a$; apply uniform block crossover with b with probability p_c
    Mutate each gene independently with probability p_m: adjacent step (p_s) or grid resampling ($1 - p_s$)
    If $x = a$, perturb one width gene
    For each block with $n_{kv} \nmid n_h$, set n_{kv} to the largest allowed divisor of n_h
    return x
end procedure

Against uniform scaling at matched validation loss, the search lowers energy per token on every accelerator and scale we ran, by a median of 8.4% across the eight rows of Table 9 and by up to 12.4% in the best single seed. Every seed of every row saves energy against its uniform baseline, and the spread across seeds is narrow, at most 5.0 percentage points on any row.

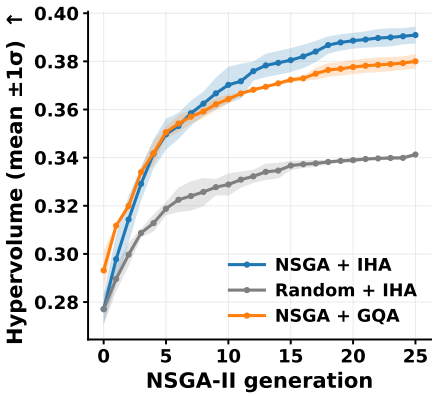


Figure 6: Search algorithm and search-space ablations across three seeds.

Latency mostly follows energy down. Time per output token falls alongside energy on seven of the eight rows, by a median of 3.6% to 9.8%, because the widths the search gives up reduce both at once. The ring at SmolLM2-135M is the exception and pays 1.9% in latency for its energy across all three of its seeds, which is the one place where the two costs pull against each other rather than together. Whether energy and latency can be traded against each other on one target is a separate question, which §3.3 takes up by varying the objective rather than the target.

To distinguish the contributions of the search algorithm and search space, Figure 6 compares three variants. NSGA + IHA uses the full method, Random + IHA replaces evolutionary search with uniform sampling, and NSGA + GQA restricts NSGA-II to GQA-compatible configurations. All variants use the elastic-KV supernet on Gemmini with SmolLM2-135M, three seeds, and 832 candidate evaluations per seed. Because the IHA and GQA spaces have different minimum-size architectures, hypervolume is computed using a common normalization box. NSGA + IHA reaches a final hypervolume of 0.391, compared with 0.341 for Random + IHA and 0.380 for NSGA + GQA, indicating benefits from both evolutionary search and the expanded architecture space.

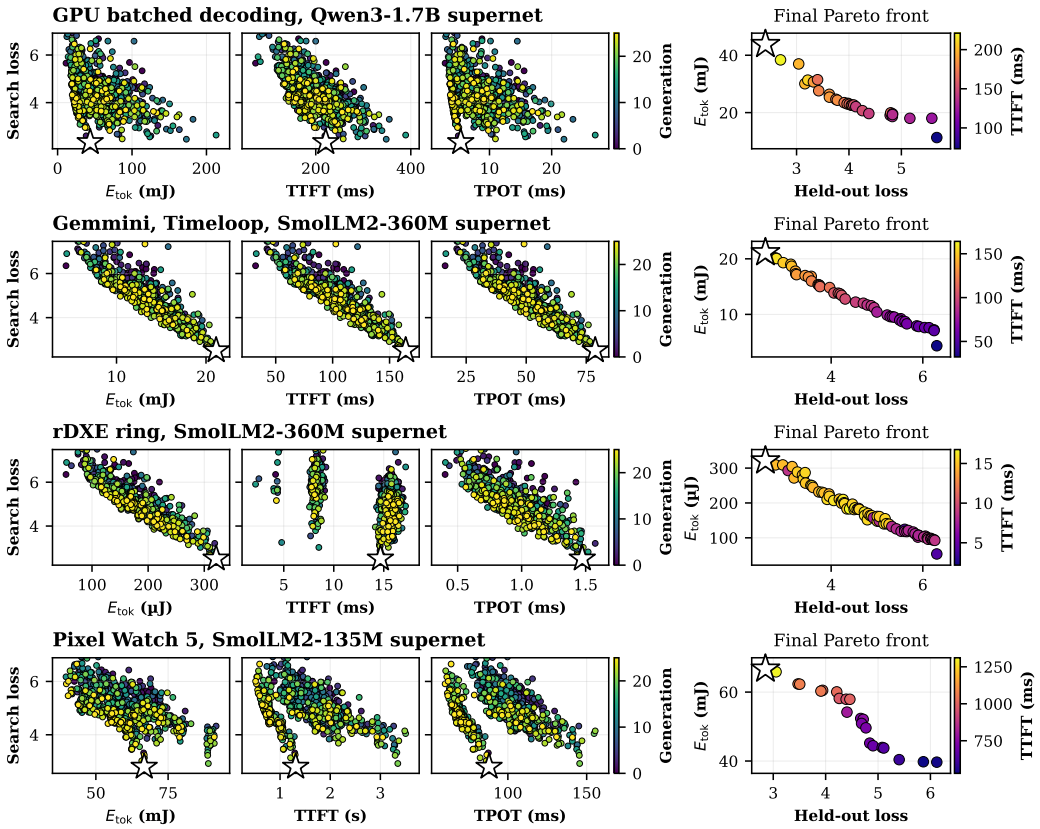


Figure 5: Search progress and final Pareto fronts across four backend classes. The three left panels show evaluated architectures colored by generation. The right panel shows fronts re-scored on held-out data, with color indicating TTFT and stars marking full-width supernet architectures.

3.2 Adapting the Architecture to the Hardware

We examine how deployment targets affect capacity allocation using a shared SmolLM2-360M supernet and search space. Searches for Gemmini, the rDXE ring, and the smartwatch all optimize held-out loss, energy per token, and time per output token. We vary the allowed increase in loss from 0% to 40% relative to the full-width supernet and, at each threshold, select the lowest-energy architecture satisfying this requirement from each seed's Pareto front. Figure 7 summarizes the resulting configurations.

Across these fronts, 86 of the 95 architectures within the loss range use different configurations across blocks, showing that the search exploits nonuniform capacity allocation. Although all three targets favor smaller MLPs at larger loss budgets, they differ most clearly in the value head dimension. The smartwatch retains full d_v throughout the range in all three seeds, while the mean normalized values for Gemmini and the ring reach approximately 79% and 89%, respectively. The smartwatch's seed range remains disjoint from each accelerator's over approximately half of the budget range, indicating a consistent difference under the same objectives.

Differences between Gemmini and the ring are generally comparable to variation across seeds, so the clearest separation in this experiment is between the smartwatch and modeled accelerator backends. This result highlights the role of target-specific cost evaluation in architecture selection. While parameter count assigns an architecture the same cost on every platform, hardware backends capture the deployment-dependent costs that guide capacity allocation.

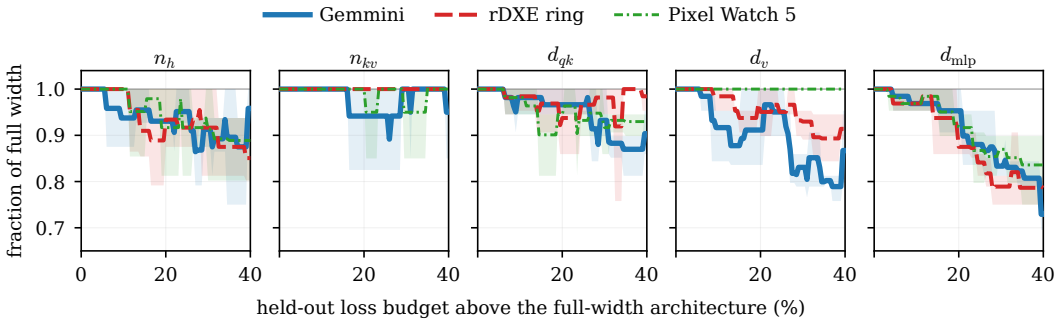


Figure 7: SmolLM2-360M capacity allocation across hardware targets. Each variable is normalized to its full-width value and averaged over five blocks. Lines and shaded bands show the mean and range across three seeds.

3.3 Architecture Selection under Deployment Objectives

Different deployment objectives can favor different capacity allocations. We compare three objective settings for SmolLM2-135M on the rDXE ring, pairing model loss with energy per token, TTFT, or TPOT. All settings use the same supernet and search space, with three seeds and 832 evaluations per seed. To compare their selected architectures under a common quality constraint, we allow at most a 40% increase in held-out loss relative to the full-width supernet. Within this bound, each search contributes the Pareto-front architecture that minimizes its hardware objective (Figure 8).

The clearest difference is between TPOT and TTFT optimization. Across seeds, TPOT-selected architectures retain 60–67% of the full query-head count on average across the network, compared with 87–93% for TTFT-selected architectures. Most of the head reduction occurs in the last three blocks, showing that the objective affects both the amount of attention capacity retained and its distribution across the network.

A regression analysis of backend costs, accounting for layer-to-chip packing, is consistent with this difference (Table 7). TPOT has a stronger fitted sensitivity to query-head count than TTFT, whereas TTFT is more sensitive to MLP width. The other four search variables have overlapping seed ranges at this loss bound, making query-head allocation the clearest observed response to changing the deployment objective.

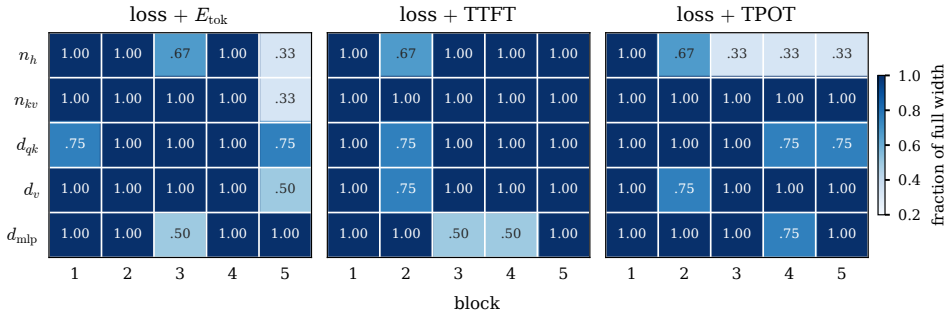


Figure 8: SmolLM2-135M configurations selected under different objectives on the rDXE ring, allowing up to a 40% loss increase. Cells show medians across three seeds of block-averaged values normalized to the full configuration.

4 Related Work

Attention design and capacity allocation. Efficient language model architectures adjust both the structure of attention and the distribution of capacity across layers. Within attention, GQA (Ainslie et al., 2023) reduces key–value storage by sharing heads across queries, while KV-Latent (Shi et al.,

2025) reduces key–value dimensions using frequency-aware rotary embeddings. At the layer level, OpenELM (Mehta et al., 2024) varies widths across the network to allocate parameters more effectively. Related choices can also be made after pretraining: SliceGPT (Ashkboos et al., 2024) and Minitron (Muralidharan et al., 2024) use structured pruning to derive smaller models from pretrained checkpoints.

Weight sharing and pretrained model adaptation. Exploring these architectural choices requires an efficient way to compare candidate quality. Once-for-all networks (Cai et al., 2020) and HAT (Wang et al., 2020) address this cost through supernets whose subnetworks share parameters, reducing the need to train each candidate independently. Flextron (Cai et al., 2024) adapts this strategy to pretrained language models, using continued training to construct nested elastic attention and MLP layers. This approach allows multiple configurations to inherit weights from a common model.

Hardware-aware search and co-design. Alongside candidate quality, architecture selection must account for the cost of deployment. Existing methods capture this cost at different levels: STAR (Thomas et al., 2025) uses parameter count and inference-cache size as efficiency objectives, while HW-GPT-Bench (Sukthanker et al., 2024) combines weight-sharing quality evaluations with hardware-cost surrogates across CPUs and GPUs. When the accelerator is also a design variable, TransCODE (Tuli & Jha, 2023) jointly searches Transformer and hardware configurations using predicted accuracy and simulated costs. LLMForge combines elasticity in head dimensions and key–value grouping with interchangeable hardware backends for target-specific architecture search. Table 1 compares the scope of these approaches.

Table 1: Comparison with representative related work.

Method	Design space	Evaluation approach	Hardware evaluation
Flextron	Heads, MLP width	Elastification + loss surrogate	GPU latency lookup table
HW-GPT-Bench	Depth, width, heads, MLP	Supernet + surrogates	CPU/GPU cost surrogates
TransCODE	Transformer + accelerator	Accuracy surrogate	Accelerator simulation
LLMForge	**IHA + MLP width**	Elastification of pretrained models	GPU measurements; ASIC models; smartwatch predictor

5 Conclusion

We presented LLMForge, a hardware-aware architecture search framework that combines Infinite-Head Attention and variable MLP widths to explore capacity allocation within and across layers. Model elastification enables candidate quality evaluation through shared weights without training each architecture separately, while interchangeable hardware backends provide the execution costs that guide multi-objective search.

Across eight model–accelerator configurations, LLMForge reduces modeled energy per token by a median of 8.4% relative to uniform scaling at matched supernet validation loss. Ablations indicate benefits from both evolutionary search and the expanded attention space, while comparisons across hardware backends and latency objectives show that deployment requirements can change the preferred allocation of model capacity. These findings support adapting language model architectures through joint consideration of model quality, target-specific execution costs, and deployment objectives.

Limitations and future work. The dedicated-training study in Appendix C assesses shared-weight ranking reliability, while full-budget training and deployment evaluation of selected architectures remain important next steps. Our search varies widths while keeping depth, residual width, and vocabulary fixed and sharing configurations within blocks. Extending these choices and allowing finer variation across layers could broaden the search space. The non-GPU backends evaluate 8-bit weights and activations, so architecture preferences depend on this numerical format. Joint architecture and precision search could account for how different formats affect model quality and hardware cost.

ETHICS STATEMENT

This work searches architectures for existing open-weight language models. It involves no human subjects, no personal data and no new data collection. The base models and the datasets used for scoring are public and are used under their own licenses. The objective optimized throughout is lower energy per token at matched quality, and the risks of the resulting models are those of the base models they are derived from.

AI USE STATEMENT

Generative AI tools were used as coding and editing assistants. They helped write and refactor experiment and analysis scripts in the code release and assisted with copy editing of the manuscript. The research questions, the experimental design, the measurements and every claim in this paper were produced and verified by the authors, who take full responsibility for its content.

REPRODUCIBILITY STATEMENT

The search grid of every base model is given in Appendix B and ships as a configuration file in the code release. Appendix C gives the supernet training recipe and the checks on its ranking quality. Appendices D, E, F and G give the measurement or modeling protocol of each backend, and Appendix H gives the search configuration. Appendix J reports the compute each experiment used. The anonymized code release at `https://anonymous.4open.science/r/llmforge_iclr_codebase-6326` contains the elastification, the search, the four backends and the scripts that produce every figure and table in this paper.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4895–4901, 2023. doi: 10.18653/v1/2023.emnlp-main.298.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM2: When Smol goes big – data-centric training of a fully open small language model. In *Second Conference on Language Modeling (COLM)*, 2025. URL `https://openreview.net/forum?id=3JiCl2A14H`.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *International Conference on Learning Representations (ICLR)*, 2024. URL `https://openreview.net/forum?id=vXxardq6db`.

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 (ICLR)*, 2020. URL `https://openreview.net/forum?id=HylxE1HKwS`.

Ruisi Cai, Saurav Muralidharan, Greg Heinrich, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. Flextron: Many-in-One flexible large language model. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, pp. 5298–5311, 2024. URL `https://proceedings.mlr.press/v235/cai24e.html`.

Yu-Hsin Chen, Tien-Ju Yang, Joel Emer, and Vivienne Sze. Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(2):292–308, 2019. doi: 10.1109/JETCAS.2019.2910232.

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.

Hasan Genc, Seah Kim, Alon Amid, Ameer Haj-Ali, Vighnesh Iyer, Pranav Prakash, Jerry Zhao, Daniel Grubb, Harrison Liew, Howard Mao, Albert Ou, Colin Schmidt, Samuel Steffl, John Wright, Ion Stoica, Jonathan Ragan-Kelley, Krste Asanovic, Borivoje Nikolic, and Yakun Sophia Shao. Gemmini: Enabling systematic deep-learning architecture evaluation via full-stack integration. In *Proceedings of the 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 769–774, 2021. doi: 10.1109/DAC18074.2021.9586216.

Sheng-Chun Kao, Suvinay Subramanian, Gaurav Agrawal, Amir Yazdanbakhsh, and Tushar Krishna. FLAT: An optimized dataflow for mitigating attention bottlenecks. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 295–310, 2023. doi: 10.1145/3575693.3575747.

Stefanos Laskaridis, Kleomenis Katevas, Lorenzo Minto, and Hamed Haddadi. MELTing Point: Mobile evaluation of language transformers. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking (MobiCom)*, pp. 890–907, 2024. doi: 10.1145/3636534.3690668.

Sachin Mehta, Mohammad Hossein Sekhavat, Qingqing Cao, Maxwell Horton, Yanzi Jin, Chenfan Sun, Iman Mirzadeh, Mahyar Najibi, Dmitry Belenko, Peter Zatloukal, and Mohammad Rastegari. OpenELM: An efficient language model family with open training and inference framework. In *ICML 2024 Workshop on Efficient Systems for Foundation Models (ES-FoMo-II)*, 2024. URL https://openreview.net/forum?id=XNMbTkxroF.

Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. Compact language models via pruning and knowledge distillation. In *Advances in Neural Information Processing Systems 37 (NeurIPS)*, pp. 41076–41102, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/4822991365c962105b1b95b1107d30e5-Abstract-Conference.html.

Angshuman Parashar, Priyanka Raina, Yakun Sophia Shao, Yu-Hsin Chen, Victor A. Ying, Anurag Mukkara, Rangharajan Venkatesan, Brucek Khailany, Stephen W. Keckler, and Joel Emer. Timeloop: A systematic approach to DNN accelerator evaluation. In *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 304–315, 2019. doi: 10.1109/ISPASS.2019.00042.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems 37 (NeurIPS, Datasets and Benchmarks Track)*, pp. 30811–30849, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/370df50ccfdf8bde18f8f9c2d9151bda-Abstract-Datasets_and_Benchmarks_Track.html.

Luohe Shi, Zuchao Li, Lefei Zhang, Baoyuan Qi, Guoming Liu, and Hai Zhao. KV-Latent: Dimensional-level KV cache reduction with frequency-aware rotary positional embedding. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1535–1550, 2025. doi: 10.18653/v1/2025.acl-long.77.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063.

Rhea Sanjay Sukthanker, Arber Zela, Benedikt Staffler, Aaron Klein, Lennart Purucker, Joerg K. H. Franke, and Frank Hutter. HW-GPT-Bench: Hardware-aware architecture benchmark for language models. In *Advances in Neural Information Processing Systems 37 (NeurIPS, Datasets and Benchmarks Track)*, pp. 60776–60834, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/6ffdbf064df51857eb802a904aaaba63-Abstract-Datasets_and_Benchmarks_Track.html.

Guanchen Tao, Junyi Luo, Shiwei Liu, Anhang Li, Gregory Kielian, Kauna Lei, Qirui Zhang, Dennis Sylvester, and Mehdi Saligane. An 11.16 μJ/token edge SLM decoder accelerator with scalable ring-based configuration for token-level pipelining in 16nm FinFET. In *2026 IEEE Custom Integrated Circuits Conference (CICC)*, pp. 1–4, 2026. doi: 10.1109/CICC65509.2026.11509572.

Armin W. Thomas, Rom N. Parnichkun, Alexander Amini, Stefano Massaroli, and Michael Poli. STAR: Synthesis of tailored architectures. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=HsHxSN23rM.

Shikhar Tuli and Niraj K. Jha. TransCODE: Co-design of transformers and accelerators for efficient training and inference. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(12):4817–4830, 2023. doi: 10.1109/TCAD.2023.3283443.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pp. 5998–6008, 2017. URL https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Hanrui Wang, Zhanghao Wu, Zhijian Liu, Han Cai, Ligeng Zhu, Chuang Gan, and Song Han. HAT: Hardware-aware transformers for efficient natural language processing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 7675–7688, 2020. doi: 10.18653/v1/2020.acl-main.686.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Jie You, Jae-Won Chung, and Mosharaf Chowdhury. Zeus: Understanding and optimizing GPU energy consumption of DNN training. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 119–139, 2023. URL https://www.usenix.org/conference/nsdi23/presentation/you.

A Experimental Setup

Table 2: Experimental setup across hardware backends. Workloads are listed as (prompt tokens, generated tokens, batch size). For Backend D the two objective sets correspond to the two base models in the order listed. Backend C also carries the three further objective sets that §3.3 compares.

Backend	Target	Workload	Objectives	Base models
A	GPU (H100)	512, 128, 64	loss, E_{tok}, TPOT	Qwen3-1.7B, Qwen3-4B
B	Gemmini, Eyeriss, FLAT	128, 32, 1	loss, E_{tok}, TPOT	SmolLM2-135M, SmolLM2-360M
C	rDXE ring	128, 32, 1	loss, E_{tok}, TPOT loss, E_{tok} loss, TTFT loss, TPOT	SmolLM2-135M, SmolLM2-360M, Qwen3-0.6B
D	Pixel Watch 5	48, 32, 1	loss, E_{tok} loss, E_{tok}, TPOT	SmolLM2-135M SmolLM2-360M

B Search Space Specification

Table 3 lists the base models and their search grids. Every layer of a block shares one choice. Each base model has five attention blocks and five MLP blocks of contiguous layers, except Qwen3-4B, whose MLP uses four. Blocks are cut as evenly as the depth allows, with the i-th boundary at $\lfloor iL/5 \rfloor$, so SmolLM2-135M falls into five blocks of six layers while SmolLM2-360M, Qwen3-0.6B and Qwen3-1.7B have blocks that differ by one layer. Qwen3-4B is the only base model for which we measured per-layer sensitivity, and both its attention and its MLP blocks follow that measurement instead of the even cut. Its attention blocks hold 1, 11, 10, 3 and 11 layers and its MLP blocks hold 1, 3, 30 and 2. Each attention block chooses n_{kv} first and then n_h among the head counts that this n_{kv} divides, which gives the (n_h, n_{kv}) pairs of Table 3. Depth, d_{model} and the vocabulary stay at the base values.

Table 3: Base models and search grids. d_{qk} and d_v share one grid and d_{mlp} takes the four quarter steps of the base width. Pairs counts the (n_h, n_{kv}) combinations in which n_{kv} divides n_h. Uniform counts the architectures whose layers share one shape, and heterogeneous counts the block-level configurations. The cache floor is the smallest key–value cache relative to the base model.

Base model	L	d_{model}	n_{kv} grid	n_h grid	Pairs	d_{qk}, d_v grid	Base d_{mlp}	Uniform	Heterogeneous	Cache floor
SmolLM2-135M	30	576	1, 3	3, 6, 9	6	16, 32, 48, 64	1536	384	8.3×10^{12}	8.3%
SmolLM2-360M	32	960	1, 5	5, 10, 15	6	16, 32, 48, 64	2560	384	8.3×10^{12}	5.0%
Qwen3-0.6B	28	1024	1, 2, 4, 8	2, 4, 8, 16	13	32, 64, 96, 128	3072	832	4.0×10^{14}	3.1%
Qwen3-1.7B	28	2048	1, 2, 4, 8	2, 4, 8, 16	13	32, 64, 96, 128	6144	832	4.0×10^{14}	3.1%
Qwen3-4B	36	2560	1, 2, 4, 8	4, 8, 16, 32	15	32, 64, 96, 128	9728	960	2.0×10^{14}	3.1%

Freeing the head dimensions is what makes the space large. Per block, Infinite-Head Attention offers 96 attention shapes at either SmolLM2 scale, 208 at Qwen3-0.6B and Qwen3-1.7B, and 240 at Qwen3-4B. Tying d_v to d_{qk} would leave 24, 52 and 60 of them, and additionally tying the head dimension to d_{model}/n_h, as grouped-query attention does, would leave 6, 13 and 15.

C Supernet Training and Ranking Quality

Table 4 lists the supernet behind every search in this paper. All five share a recipe and differ only in the learning rate, which a sweep at each scale selected. Every supernet trains with the sandwich rule of §2.2 for 2,500 steps of 16,384 tokens, which is 41.0M tokens in total, using AdamW with betas 0.9 and 0.95, weight decay 0.1 and gradient clipping at 1.0. Each run warms up for 200 steps and then decays the learning rate to zero on a cosine over the remainder. Sampled subnetworks are distilled from the frozen base model at a temperature of 2.0 with the distillation term at unit weight, and the rotation angles of the key–value groups train in their own AdamW group at a learning rate of

0.05 without weight decay. The learning rates are an order of magnitude below the values an earlier round of this work used, and the searches in §3 all run on the supernets trained at the rates given here.

Table 4: Supernet training recipes, one row per base model. Every run takes 2 sequences of 1,024 tokens per micro-batch with 8 gradient accumulation steps, which is 16,384 tokens per step and 41.0M tokens over 2,500 steps. The mixture takes 40% web, 25% code, 20% math and 15% retrieval text in every row, drawn from the FineWeb-Edu family for SmolLM2 and from the corresponding Qwen3 sources at the larger scales.

Base model	Learning rate	Checkpoint	Distillation	Teacher
SmolLM2-135M	3.6×10^{-4}	step 2500	$\tau = 2.0$, weight 1.0	frozen base
SmolLM2-360M	2.1×10^{-4}	step 2500	$\tau = 2.0$, weight 1.0	frozen base
Qwen3-0.6B	2×10^{-4}	step 2500	$\tau = 2.0$, weight 1.0	frozen base
Qwen3-1.7B	1×10^{-4}	step 2500	$\tau = 2.0$, weight 1.0	frozen base
Qwen3-4B	8×10^{-5}	step 2500	$\tau = 2.0$, weight 1.0	frozen base

What the slicing rules are worth before training. Table 5 slices the untrained SmolLM2-135M, whose temperatures and rotation angles are still zero, under each rule and under a naive counterpart, and scores every slice on the held-out documents. The naive rotary order keeps the highest-frequency pairs, the naive head selection keeps a prefix of query heads that still read their own groups, and the naive group reduction keeps the first group of every merged bundle instead of their mean. The interleaved rotary order costs 3.4 to 5.2 nats less than the high-frequency prefix at d_{qk} of 32 and 48 and beats the median random order, but at $d_{qk} = 16$ every order is poor and one of eight random orders is 2.5 nats better. Balanced head selection costs 1.4 to 3.3 nats less than a head prefix. Mean pooling beats keeping one group only when all query heads remain, because a smaller head count keeps mostly heads that belong to the first group. Combined over all 384 uniform shapes, the naive rules give a median loss 0.02 nats lower than ours, but this compares broken models. Only 5 of these slices come within 3 nats of full width, and our rules give the lower loss on 4 of them by a median 3.4 nats, while the others score 13 to 15 nats, above the 10.8 nats of a uniform prediction over the vocabulary. Training changes the picture. On the trained Qwen3-1.7B supernet without key–value group reduction, every knob sliced in its trained order beats the best of 32 random orders at 9 of 10 widths, by up to 1.1 nats, and loses by 0.021 nats at the tenth. This shows that training makes the ordered slices good, but only a trained ablation can show that the interleaved order is the better order to train with. The temperatures and the rotation angles act only after training, so this comparison cannot test them.

Table 5: Held-out loss of slices of the untrained SmolLM2-135M under each slicing rule and its naive counterpart. The full-width model scores 2.63. The random columns report the median and the best of eight random pair orders.

Rule	Slice	Ours	Naive	Random, median	Random, best
Rotary pair order	$d_{qk} = 48$	3.16	8.31	4.73	3.89
	$d_{qk} = 32$	5.49	8.84	6.82	5.40
	$d_{qk} = 16$	8.68	8.94	8.36	6.22
Head selection	$n_h = 6$	4.87	8.21		
	$n_h = 3$	10.3	11.7		
Group reduction	$n_{kv} = 1, n_h = 9$	10.9	13.9		
	$n_{kv} = 1, n_h = 6$	12.9	12.7		
	$n_{kv} = 1, n_h = 3$	13.1	11.7		

Whether slice scores rank models that were actually trained. The search ranks candidates by a shared-weight slice score, and the question this appendix has to answer is whether that ranking survives dedicated training. We trained 20 architectures to convergence under one recipe and compared each one's held-out loss after training against the slice score the supernet had assigned it. The set spans what the search explores: three points from the searched fronts, budget-matched uniform controls, the heterogeneous and uniform pairs used in the conversion study, six points placed to span the

key–value axis, and four uniform grouped-query baselines reduced physically rather than by slicing. Two of the twenty are the same architecture trained under two data seeds and carry an identical slice score, so that pair is excluded and 189 of the 190 ordered pairs remain.

Slice scores rank the trained models at a Kendall τ of $+0.92$ and a Spearman ρ of $+0.98$, with 181 of the 189 pairs concordant. The ranking holds at the resolution the search works at. All eight inversions sit below 0.115 nats of predicted separation, and above 0.20 nats the ordering is correct on every one of the 147 pairs.

The compression the slice introduces is local rather than global. Across the twenty points the full spread contracts only from 1.88 to 1.53 nats, which is 81% of it, and pairs the slice separates by 0.50 nats or more keep 89% of their separation after training. What collapses is the near ties, which is why the slice is usable as a selection signal at the resolution the search works at and not as a calibrated estimate of trained loss. Where two candidates fall inside 0.10 nats of each other the crowding rule of the front, rather than the slice, is what decides between them.

Scoring protocol. Every candidate is scored on 1,024-token windows cut from a fixed pool of held-out documents. The objective the search minimizes reads the first 32 documents of that pool, and the final front is re-scored on the next 64, which are disjoint from the ones the search saw. Every loss reported in this paper is the re-scored one.

D GPU Measurement Protocol

Backend A measures a candidate on one H100 in bf16 rather than modeling it. Each candidate is instantiated with a preallocated key–value cache and native grouped-query attention, so the measured shape is the one the search asked for rather than a padded stand-in. The workload is 512 prompt tokens, 128 generated tokens and a batch of 64. Decode runs under CUDA graphs, which removes launch overhead from the window being measured. Every candidate is measured once as a warmup and then three times for record, with prefill and decode grouped rather than interleaved, a settling pause of 1.5 seconds between them, a minimum window of 2 seconds and a decode window of 6 seconds. Grouping and the settling pause both come from a pilot in which decode windows opened immediately after prefill read high, because the device had not returned to its idle power state. Measurements are cached by architecture and by measurement setting, so a repeated candidate is not remeasured.

E Timeloop Substrates

Backend B prices a candidate on three modeled substrates, a Gemmini-style systolic array, an Eyeriss-style spatial array and a FLAT-style edge accelerator. Each transformer layer is expressed as seven GEMMs, which are mapped and costed separately for the prefill pass and for one decode step, and adjacent operators are fused where the substrate allows it. The workload is 128 prompt tokens, 32 generated tokens and a batch of one, matching the edge setting the ring accelerator also targets. One property of these substrates bounds what any objective study on them can show. Fitted on the same candidates, the elasticities of energy per token, time to first token and time per output token with respect to the five widths agree to three decimals on Gemmini, at a cosine of 1.000 between their coefficient vectors, so on these substrates the three costs are one cost and no choice among them can change what a search returns. Table 6 gives the three substrates as they are modeled. All three are read at 8-bit weights and activations over an LPDDR4 interface and evaluated at a 16 nm technology node, so what separates them is the shape of the array and the memory around it rather than the process they are costed on.

F Ring Accelerator Co-Search

Backend C differs from the others in searching a chip configuration alongside the architecture. Every candidate is swept over a grid of 45 chip configurations, the product of three multiplier widths per vector core, three caps on ring size and five on-chip weight memory sizes. For each configuration, balanced packing assigns contiguous layers to ring stages, the ring simulator returns time to first token, time per output token and energy per token, and configurations outside the area and power

Table 6: The three modeled substrates of Backend B. The array is the spatial extent over which one GEMM is mapped, and the on-chip memory is the buffer between the array and DRAM. Every substrate reads 8-bit weights and activations over LPDDR4 at a 16 nm node.

Substrate	Array	On-chip memory	Per-element storage
Gemmini-style	16×16 systolic	256 KB scratchpad, 4 banks	one accumulator and one register
Eyeriss-style	14×12 spatial	128 KB global buffer, 32 banks	12, 192 and 16 entry spads
FLAT-style edge	32×32	200 KB global buffer, 32 banks	32 entry accumulator, one register

envelope are discarded. The run keeps the subset that is not dominated over those three costs, which is nine configurations for a typical candidate, and the search then prices the candidate on the cheapest of them for whichever cost it is optimizing. Two of the three chip dimensions saturate. Every candidate in every run selects the smallest multiplier width and the smallest weight memory, so the dimension that co-adapts to the architecture is the cap on ring size, and the realized chip count follows from that cap together with how the layers pack. The chip the sweep varies is described by two numbers, the multipliers it holds and the weight memory that feeds them. The reference chip holds 2,048 multipliers and 3 MB of weight memory, organized as vector cores of 16, 32 or 64 multipliers each, so the smallest setting gives 128 cores of 24 KB and the largest gives 32 cores of 384 KB. Layers are assigned to chips by balanced packing, and a chip whose weight memory cannot hold the layers it is given reloads them, which the simulator charges for. Chips are connected in a ring and pass one embedding vector per hop, so the communication energy of a prefill scales with the number of active chips and the prompt length, and the communication energy of a decode step scales with the number of active chips alone. This is what makes provisioning more chips lower time to first token without lowering time per output token, and it is why the ring is the one backend in this paper where the search chooses hardware as well as architecture.

G On-Device Predictor

Backend D is a predictor rather than a device in the loop, because a search of this size cannot be run on a watch. The measurements behind it come from an INT8 C inference runtime on a Pixel Watch 5 at a workload of 48 prompt tokens, 32 generated tokens and a batch of one. The predictor is three independent XGBoost regressors, one per cost, each fitted in log space on 110 architecture-only features. It reads shape alone, with no temperature, battery, voltage or measured timing among its inputs.

The dataset holds 2,000 measured per-layer architectures under a fixed grouped split of 1,600 for fitting, 200 for validation and early stopping, and 200 held back as a test set the fit never sees. On that test set the predictor reaches a mean absolute percentage error of 14.0% on energy per token with a rank correlation of 0.94, 16.5% and 0.92 on TTFT, and 15.5% and 0.84 on time per output token. Its Kendall τ_b is 0.79, 0.78 and 0.67 on the three costs, and it orders 89.6%, 88.9% and 83.3% of the 19,900 test pairs correctly. Every figure is the mean over three training seeds, whose errors differ by at most 0.22 percentage points.

Each regressor uses depth-3 trees, a learning rate of 0.03, L2 regularization of 5, row and column subsampling of 0.85 and 0.9, and up to 1,500 trees, stopping early after 50 rounds without improvement in validation log error. The search uses the checkpoint of the seed with the lowest mean validation error, chosen before the test set was evaluated, and its test errors differ from the three-seed means by at most 0.04 percentage points. Energy per token here is the dynamic energy of the whole request, prefill included, divided by the 32 output tokens, with the idle baseline subtracted.

H NSGA-II Configuration

A candidate is encoded as one choice per knob per block, which the variation operators treat as a flat list of categorical genes. Every search keeps a population of 32, produces 32 offspring per generation and runs for 25 generations, which is 832 evaluations including the initial population. Parents are chosen by binary tournament, offspring are produced by uniform crossover at a rate of 0.9 and then mutated one gene at a time, and survivors are taken by non-dominated sorting

Table 7: How three deployment costs price width on the ring, and what share of its width reduction each search gives up on each knob. Every cost is fitted on the logs of the five knobs, each averaged over the layers, with the layer-to-chip packing entered as a fixed effect, over the 4,897 of 4,992 candidates the six unconstrained runs evaluated that fall in packing levels holding at least 40 of them, at R^2 0.977, 0.982 and 0.996 in column order. Every difference excludes zero under a bootstrap over runs. Shares are taken inside a 40% loss increase and given as the range over three seeds.

	Elasticity			Difference from E_{tok}		Share of reduction (%)		
Knob	E_{tok}	TTFT	TPOT	−TTFT	−TPOT	E_{tok}	TTFT	TPOT
n_h	0.256	0.206	0.642	+0.050	−0.386	19 to 33	27 to 30	34 to 47
n_{kv}	0.182	0.026	0.009	+0.156	+0.173	0 to 28	4 to 20	7 to 22
d_{qk}	0.123	0.063	0.033	+0.060	+0.090	10 to 16	14 to 20	17 to 30
d_v	0.086	0.042	0.025	+0.044	+0.061	19 to 32	14 to 24	8 to 24
d_{mlp}	0.585	0.635	0.317	−0.050	+0.268	10 to 23	14 to 34	3 to 9

with the crowding distance breaking ties inside a front. Where a search carries a bound, feasible candidates take precedence over infeasible ones under constrained domination and the magnitude of the violation orders the infeasible ones among themselves, so a population that starts outside the feasible region is still guided into it. Every search in this paper runs at three seeds, and no result rests on a single one unless the text says so. Hypervolume is computed against a reference point fixed per backend and scale, so the values in Table 9 compare down a column and not across them.

I Additional Results

This appendix collects the per-backend and per-scale numbers that §3 reports only in aggregate, together with the two block-level views the main text has room for only at one scale.

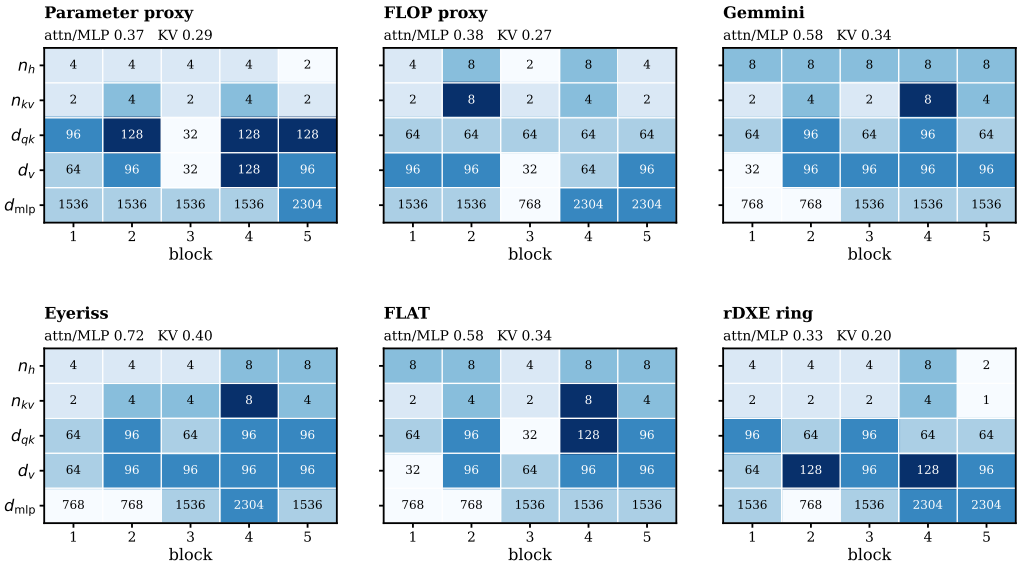


Figure 9: Block allocation of every target behind §3.2 at Qwen3-0.6B, one cell per block and per knob, with colors normalized per knob. The three substrates that stream weights from off chip sit between 0.58 and 0.72 on the attention to MLP ratio and the ring accelerator sits at 0.33. The two proxies fall at 0.37 and 0.38 on this one architecture each, closer to the ring than to the substrates, which the band averages of Table 8 do not show. A single lowest-cost pick is not the band.

Table 9 gives the record behind §3.1 for every backend with a measured uniform baseline, including the hypervolume of each front and the saving in time per output token.

Table 8: Every knob of the winners whose non-embedding parameter count falls in 0.40 to 0.65 of the full-width base, as a fraction of that base, on the five-knob elastic-KV supernets. Brackets give the range over three search seeds. The two proxies ran one seed each. The GPU has no elastic-KV run and the smartwatch's elastic-KV searches are not summarized here, so neither appears. On n_h, d_{mlp} and the attention to MLP ratio the ring's interval meets no other target's at any of the three scales, while the cache and n_{kv} separate at SmolLM2-135M and Qwen3-0.6B and overlap at SmolLM2-360M.

Target	n_h	n_{kv}	d_{qk}	d_v	d_{mlp}	KV	attn/MLP
SmolLM2-135M							
Gemmini	0.69 [0.65, 0.75]	0.77 [0.76, 0.79]	0.73	0.72	0.53 [0.52, 0.54]	0.58 [0.54, 0.61]	0.99
Eyeriss	0.68 [0.62, 0.72]	0.79 [0.72, 0.82]	0.74	0.73	0.53 [0.52, 0.54]	0.59 [0.55, 0.64]	0.99
FLAT	0.69 [0.65, 0.71]	0.80 [0.73, 0.87]	0.75	0.75	0.53 [0.52, 0.55]	0.61 [0.56, 0.69]	1.01
rDXE ring	0.44 [0.42, 0.45]	0.65 [0.62, 0.66]	0.75	0.79	0.57 [0.56, 0.58]	0.50 [0.49, 0.51]	0.67
Parameter proxy	0.71	0.79	0.72	0.72	0.51	0.56	1.04
FLOP proxy	0.71	0.91	0.75	0.67	0.53	0.64	1.01
SmolLM2-360M							
Gemmini	0.70 [0.67, 0.76]	0.90 [0.84, 0.93]	0.78	0.71	0.52 [0.52, 0.52]	0.67 [0.63, 0.72]	1.07
Eyeriss	0.68 [0.62, 0.76]	0.88 [0.86, 0.89]	0.76	0.71	0.50 [0.50, 0.51]	0.66 [0.65, 0.67]	1.09
FLAT	0.71 [0.67, 0.76]	0.89 [0.86, 0.93]	0.77	0.71	0.52 [0.51, 0.54]	0.68 [0.64, 0.72]	1.08
rDXE ring	0.46 [0.43, 0.49]	0.84 [0.81, 0.88]	0.73	0.76	0.57 [0.56, 0.58]	0.63 [0.58, 0.68]	0.73
Parameter proxy	0.66	0.91	0.76	0.66	0.53	0.65	0.98
FLOP proxy	0.74	0.96	0.73	0.66	0.52	0.67	1.08
Qwen3-0.6B							
Gemmini	0.49 [0.48, 0.49]	0.63 [0.60, 0.65]	0.69	0.71	0.60 [0.60, 0.60]	0.45 [0.44, 0.47]	0.63
Eyeriss	0.49 [0.48, 0.50]	0.62 [0.60, 0.64]	0.69	0.71	0.60 [0.59, 0.60]	0.45 [0.44, 0.46]	0.63
FLAT	0.51 [0.50, 0.52]	0.64 [0.64, 0.65]	0.67	0.72	0.60 [0.59, 0.62]	0.46 [0.44, 0.47]	0.64
rDXE ring	0.39 [0.36, 0.40]	0.52 [0.50, 0.53]	0.67	0.76	0.67 [0.66, 0.67]	0.37 [0.36, 0.39]	0.46
Parameter proxy	0.49	0.57	0.68	0.78	0.62	0.42	0.62
FLOP proxy	0.53	0.63	0.58	0.69	0.62	0.40	0.58

Table 9: Held-out hypervolume and median saving at matched loss against uniform scaling, in percent, on the five-knob elastic-KV space. A saving is the median over the points of the uniform front, and parentheses give the range over three search seeds.

Backend	Base model	Baseline	HV, LLMForge	HV, baseline	Energy saved	Latency saved
Gemmini	SmolLM2-135M	uniform	0.337	0.289	+6.6 (5.8, 6.7)	+6.6 (6.0, 6.8)
Gemmini	SmolLM2-360M	uniform	0.384	0.324	+9.4 (7.2, 10.9)	+9.8 (8.2, 11.8)
Eyeriss	SmolLM2-135M	uniform	0.334	0.289	+6.5 (6.3, 7.2)	+6.5 (6.1, 7.5)
Eyeriss	SmolLM2-360M	uniform	0.381	0.324	+8.7 (7.4, 12.4)	+8.0 (6.9, 12.5)
FLAT	SmolLM2-135M	uniform	0.337	0.289	+7.2 (7.0, 7.2)	+7.5 (7.1, 7.5)
FLAT	SmolLM2-360M	uniform	0.381	0.324	+8.7 (7.4, 11.1)	+9.4 (6.9, 11.0)
rDXE ring	SmolLM2-135M	uniform	0.383	0.350	+8.6 (7.6, 8.7)	−1.9 (-1.9, -1.4)
rDXE ring	SmolLM2-360M	uniform	0.424	0.370	+8.2 (7.3, 11.3)	+3.6 (1.0, 3.7)

J Compute and Code

Every experiment in this paper ran on a single H100. Training the five supernets is the one-time cost the method is built around, at 2,500 steps and 41.0M tokens each, or 205M tokens in total. Everything after that is search. The 129 searches behind this paper evaluate 100,736 candidates between them and account for roughly 78 hours of wall time, which is dominated by the ring accelerator and the Timeloop substrates at 48 and 45 searches. A search costs between 15 minutes and 45 minutes depending on the backend, since the cost model rather than the supernet is what varies: scoring a candidate against the ring sweeps 45 chip configurations, while the smartwatch predictor answers in milliseconds. Held-out loss is cached by architecture and by scoring setting and is shared across searches on the same supernet, so a candidate seen by an earlier search is not

rescored. The anonymized code is available at `https://anonymous.4open.science/r/llmforge_iclr_codebase-6326`.