CARTA: TRAINING-FREE LLM-DRIVEN TENSOR MAPPING FOR SPATIAL ACCELERATORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tensor mapping coordinates parallel execution with data reuse across an accelerator's memory hierarchy. Soter's per-task RL Tuner relearns it for each layer–accelerator pair. In CARTA, a frozen LLM given a short strategy text picks spatial allocations, loop priorities, and tile caps for Soter's deterministic mapping builder, and we ask what transfers across accelerators. On identical problems with an energy-delay product (EDP) objective, the Tuner has $1.85\times$ CARTA's EDP at batch 16 across three independent runs with DeepSeek as the LLM and $1.61\times$ with pinned open-weight Qwen3-14B. However, random search beats this weak incumbent on the main suite by $1.66\times$ at 500 evaluator calls. At 200 evaluator calls, random search has $1.21\times$ DeepSeek CARTA's EDP, while a Gaussian-process (GP) surrogate with a structured encoding and an LLM-free annealer seeded with the same text each have $1.06\times$. These margins lose support under workload resampling, reach parity at 500 calls, and reverse with Qwen3-14B, which both controls beat ($0.92\times$). Porting changes about 30 lines of strategy text without retraining, but writing them costs at least 3,000 diagnostic evaluator calls per port plus expert reading. Without the text, DeepSeek loses to the Tuner on both held-out ports. With it, Tuner/CARTA EDP is $1.13\times$ on GemminiLike and $1.02\times$ on NVDLA-Like, a tie, and the annealer has 1.03–$1.05\times$ DeepSeek's EDP. Qwen3-14B fails the port prediction registered before the port's agent runs, with $1.45\times$ the annealer's EDP on NVDLALike. The strategy text carries the port, and the model determines the increment, 1.03–$1.06\times$ for DeepSeek and negative for Qwen3-14B. We also measure how budget accounting, baseline alignment, and run selection inflate an LLM optimizer's apparent advantage.

1 INTRODUCTION

Spatial accelerators run tensor computations on processing-element arrays and a memory hierarchy, and their efficiency depends on the mapping: which loops run in parallel and which tiles stay in local memory. Parallelism alone is insufficient because reloading a tile costs energy, so a mapping must coordinate spatial allocation, tiling, and loop order under hardware constraints.

Mappers draw mapping knowledge from different sources, each with a cost. Hand-written templates need an expert to revise the reuse strategy for each new target. Knowledge-free search, such as random sampling or annealing, must find useful structure through evaluator calls. Per-task learned proposers pay in training, since Soter's RL Tuner trains a new policy for every layer–accelerator pair (Wang et al., 2024). A frozen LLM can reuse mapping knowledge without training. However, it is unclear whether transferable knowledge lives in the model or in the text it is given.

CARTA replaces the trained policy with a frozen LLM over Soter's deterministic mapping builder and analytical evaluator. The builder reduces a mapping to a few decisions compact enough to write as text, namely a spatial split at each level, a loop-priority order, and tile caps. The LLM proposes decisions, the builder completes each into a legal mapping, and the evaluator returns feedback (Figure 1). CARTA rests on three principles. First, it separates strategy from legality: the proposer chooses decisions and the builder enforces hardware constraints. Second, it transfers the strategy as 10–40 lines of text per accelerator that link hardware structure to mapping choices, so the search loop and model weights stay fixed across targets. For example, the Simba text recommends parallelizing rows at both levels to multicast weights. Third, it tests the strategy apart from its proposer,

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.

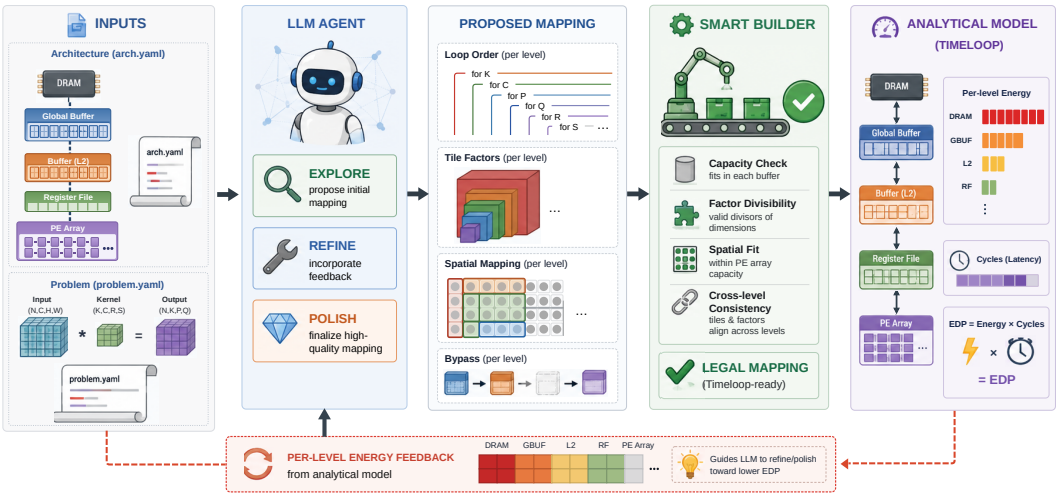

Figure 1: CARTA separates strategy selection from mapping legality. A frozen LLM uses the layer, architecture, and a strategy text of 10–40 lines per accelerator to propose spatial allocations, loop priorities, and tile caps. Soter's builder enforces capacity, divisibility, fanout, and cross-level constraints. Timeloop returns EDP and per-level energy. The prompt contains no reference mapping or target EDP. Porting changes the strategy text and architecture context. Model weights stay fixed.

as a strategy-seeded annealer given the same text separates supplied knowledge from the LLM's increment.

The fair Tuner optimizes energy-delay product (EDP) on identical problems, and at batch 16 it has $1.85\times$ CARTA's EDP across three independent DeepSeek runs, 1.77–$1.85\times$ across three Tuner seeds, and $1.61\times$ with Qwen3-14B. However, random search also beats this weak incumbent on the main suite, so controls decompose the gain. At 200 evaluator calls, random search has $1.21\times$ DeepSeek CARTA's EDP, while a Gaussian-process (GP) surrogate with a structured encoding and the strategy-seeded annealer have $1.06\times$. These margins lose support under workload resampling, reach parity at 500 calls, and reverse with Qwen3-14B, which both controls beat ($0.92\times$).

On held-out GemminiLike and NVDLALike, the unported DeepSeek agent loses to the fair Tuner. A strategy text for each brings Tuner/CARTA EDP to $1.13\times$ and $1.02\times$ but costs at least 3,000 diagnostic evaluator calls per port plus expert reading. There the annealer has $1.03\times$ and $1.05\times$ DeepSeek's EDP, whereas Qwen3-14B fails the prediction registered before the port's agent runs, with $1.45\times$ the annealer's EDP on NVDLALike. Thus the strategy text carries the port, and the model determines the increment, 1.03–$1.06\times$ for DeepSeek and negative for Qwen3-14B.

We make the following contributions:

- We decompose CARTA's gain into supplied strategy text and model increment. The strategy-seeded annealer is within $1.06\times$ of DeepSeek at 200 evaluator calls and 1.03–$1.05\times$ on the ports, reaches parity at 500, and beats Qwen3-14B.
- We design CARTA, a frozen-LLM mapper over an existing builder. Given the strategy text, it beats the fair Tuner, a weak incumbent on the main suite, on the three main accelerators and, with DeepSeek, on GemminiLike. On NVDLALike, DeepSeek ties it and Qwen3-14B loses.
- We port CARTA to two held-out accelerators and expose the transferred strategy, its development cost, and its model-dependent benefit, including a failed registered prediction.
- As a secondary contribution, we measure how budget accounting, baseline alignment, run selection, and workload uncertainty change an LLM optimizer's apparent advantage (Table 2).

2 Background and Motivation

Spatial accelerators. A spatial accelerator combines processing elements (PEs), local buffers, and shared memory, and each level offers a *spatial fanout*, its parallel capacity. The Eyeriss

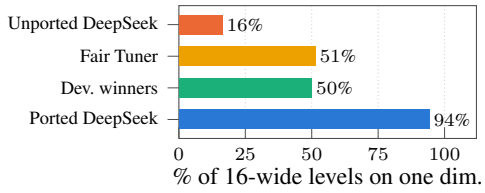

Figure 2: The strategy text makes DeepSeek give one dimension the full 16-wide GemminiLike fanout, as the development winners do, instead of splitting it. Bars: share of 16-wide spatial levels (Scratchpad, Accumulator) in best mappings held by one dimension, for unported (3 runs, one partial) and ported DeepSeek (3 runs) and the fair Tuner on the 20 test layers, and random-search winners on the 10 development layers. Mean dimensions per level: 2.30, 1.67, 1.60, 1.09.

model (Chen et al., 2016) has 256 PEs with fanouts 16×16 (the chip has 168). Simba (Shao et al., 2019) has 1024 PEs with fanouts 16×64, and the TensorCore model (NVIDIA, 2018) has 1024 PEs with fanouts 4×256. A mapping decides which dimensions fill these fanouts and which tiles each level holds. Our models, shared by all methods, enforce capacities, fanouts, and stored tensors but relax the published dataflows, including Eyeriss's row-stationary constraint.

Tensor mapping. A matrix multiplication has row dimension N, reduction dimension C, and output-channel dimension K. Parallelizing N lets rows share weights, while parallelizing K lets output channels share inputs. A convolution adds output-spatial dimensions P, Q and filter dimensions R, S, with N as batch. At each memory level, a *tensor mapping* specifies spatial factors that divide the fanout among dimensions, temporal tile factors that set the work done before a tile is replaced, and a loop order that decides which data stay reusable. These decisions are coupled: a larger tile improves reuse only if it fits, the spatial allocation changes which tiles fit, and the factors must cover every dimension within the fanouts and buffers. We minimize energy-delay product, $\mathrm{EDP} = E_{\mathrm{total}} T_{\mathrm{cycles}}$, which charges data movement and latency.

Soter's builder and evaluator. Soter's smart mapping builder (Wang et al., 2024) turns a compact decision set of spatial allocations, a priority permutation for greedy buffer filling, and optional tile caps into a complete mapping. It enforces capacity, divisibility, and cross-level consistency (App. G). Timeloop (Parashar et al., 2019) scores the loop nest analytically (access counts for energy, loop-nest analysis for cycles). CARTA replaces only the proposer.

Motivation: reuse knowledge before learning a policy. Per-task training is a weak incumbent on the main suite. The fair Tuner trains a policy per layer–accelerator pair with 320 evaluator calls, yet has $1.66\times$ the EDP of random search with 500 evaluator calls (Table 1, Figure 4).

The held-out GemminiLike accelerator exposes the missing knowledge. Lacking strategy text, the unported agent splits a 16-wide fanout across 2–3 dimensions (Figure 2). The development winners instead give one dimension the full 16 at half of the spatial levels (Scratchpad over N or Accumulator over C, each in 5 of 10 layers). The fault is accelerator-level, not layer-level. Its diagnosis fits in a sentence, and the correction, about 30 lines of strategy text, transfers to the held-out test layers. We therefore spend evaluator calls testing supplied strategy text rather than relearning it per task.

3 CARTA Framework

CARTA replaces Soter's RL Tuner with a frozen LLM (Figure 1). Each iteration turns a text proposal into a mapping through the builder, evaluates it, and adds the result to the search context. The LLM chooses strategy, and the builder supplies legality.

3.1 Architecture-Aware Prompting

The prompt separates general mapping rules, a per-accelerator strategy text (the addendum in the code and appendix), and the search context. Porting changes the strategy text and the architecture-derived context.

General mapping rules. A shared system prompt describes spatial allocation, tiling, reuse, memory energy costs, and tile caps. Each call returns five decision sets as JSON. It carries no per-accelerator strategy, although its GEMM block names TensorCore's SMEM level.

Strategy text. The strategy text, 10–40 lines per accelerator, connects the target's hierarchy to recommended decisions. The Simba text lists four strategies, including N-spatial at both fanouts for weight multicast, a $C \times K$ split at the inner fanout with P, Q caps at Registers, and C, K-first priorities at WeightBuffer. The LLM applies them per layer, and the builder completes its decisions. We wrote the port texts from development mappings (Sec. 5.4). The Eyeriss strategy text in the reported runs misstates the fanouts as 14 and 168 instead of 16 and 16.

Search context. The context supplies dimensions, hardware constraints, spatial menus (candidate allocations per level), phase instructions, and the best mappings with EDP and per-level energy. It contains no baseline mapping or target EDP for the test case. Knowledge from development mappings enters through the strategy text.

3.2 Three-Phase Search Strategy

CARTA spends 40 LLM calls per layer on exploration, refinement, and polishing. App. G gives the procedure and prompt schema (Figure 12, Algorithm 1, Table 23), and Sec. 4 the budget accounting. *Exploration* proposes output-stationary, weight-stationary, and multidimensional spatial strategies and changes strategy on stagnation. *Refinement* edits promising configurations with distinct spatial allocations. *Polishing* makes local changes to priorities, tile caps, and spatial allocations. In the first two phases, a repeated decision set is randomly perturbed once and skipped if still a duplicate. Our experiments evaluate this loop as a whole, not each phase or repeated LLM calls.

4 Experimental Setup

Accelerators and workloads. The main suite uses the Eyeriss, Simba, and TensorCore models (Sec. 2), and Timeloop's analytical `timeloop-model` scores every candidate without a cycle-accurate simulator. EDP comes from relaxed models, not hardware with published dataflows. CARTA and the search controls share the builder and evaluator, whereas the fair Tuner uses Soter's serializer with power-of-two spatial factors. The 90 layers come from 10 DNN workloads spanning standard, transposed, depthwise, group, and pointwise convolution and GEMM: ResNet (He et al., 2016), ResNeXt, DenseNet (Huang et al., 2017), VGG (Simonyan & Zisserman, 2015), UNet (Ronneberger et al., 2015), GPT-1, GPT-3 (Brown et al., 2020), BERT (Devlin et al., 2019), DeepBench (Baidu Research, 2017), and MobileNet (Howard et al., 2017). Batch sizes $B \in \{1, 16, 64\}$ give 810 cases over the accelerators, and the main search comparison uses the 270 batch-16 cases (layers in App. B).

Models. The primary backend is frozen DeepSeek-Chat, with 40 LLM calls per layer. App. B gives sampling, the Qwen3-14B server and seeding. DeepSeek ran the suite four times: a canonical 810-case sweep, a call-traced batch-16 run, and two September batch-16 runs (2×270 cases). The earlier builder revision (App. C) rebuilds all 270 traced winners exactly and allows partial fanout use. However, the current revision, used by the September runs and every control, requires spatial factors to divide the remaining fanout (powers of two here).

Baselines. The fair Tuner (Wang et al., 2024) optimizes EDP on CARTA's exact problems and trains a Transformer policy per layer–accelerator pair for 10 epochs of 32 samples (320 evaluator calls). The latency-directed Tuner optimizes latency and folds attention batches differently, so it only illustrates the corrections. Search controls comprise random search, simulated annealing, a GAMMA-style genetic algorithm (GA) (Kao & Krishna, 2020), TPE and GP surrogates, and a strategy-seeded annealer. Flat surrogate encodings treat decisions as categories, whereas structured ones expose mapping structure. The strategy-seeded annealer anneals without an LLM from templates transcribed from the strategy text. Controls with structural encodings or strategy text are *structured search*, and those with neither are *knowledge-free search*. All inherit the builder's constraints. We do not compare with native mapping tools (Soter-suite baselines: App. C).

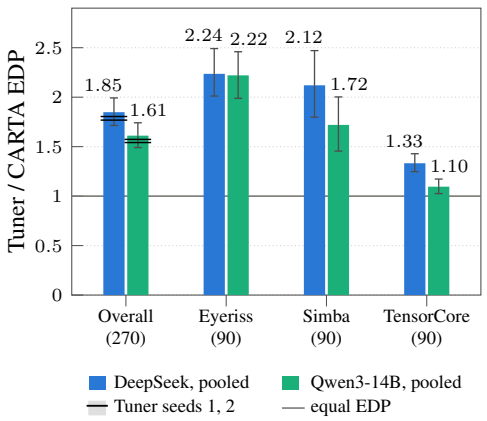

Figure 3: A frozen LLM given the strategy text beats the fair Tuner. Bars show geomean fair-Tuner EDP / CARTA EDP (above 1 favors CARTA) with layer-bootstrap 95% intervals, for DeepSeek pooled over three runs and pinned Qwen3-14B over 3 repetitions. The Tuner uses 320 evaluator calls per layer and hard-coded seed 42. Black rules on the Overall bars mark seeds 1 and 2, which give $1.80\times$ and $1.77\times$ against DeepSeek and $1.57\times$ and $1.54\times$ against Qwen.

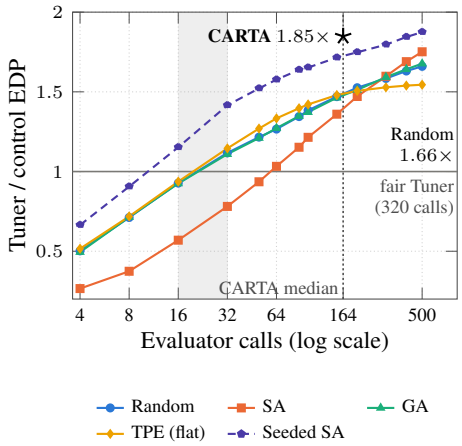

Figure 4: The fair Tuner is a weak incumbent. Random search passes it at 16–32 calls (gray band), with $0.93\times$ at 16 and $1.12\times$ at 32, and beats it $1.66\times$ at 500. Curves show geomean fair-Tuner EDP (seed 42) / best-so-far EDP of each knowledge-free control and the strategy-seeded annealer over 270 cases and 3 seeds against valid evaluator calls. The control wins above the parity line, and the dotted line marks CARTA's median 164 calls. The star marks CARTA itself (pooled DeepSeek, $1.85\times$) at its median call count. Controls stopped at 473–499 valid calls count their last best (App. C).

Budget. A new decision set costs an evaluator call even if it rebuilds a seen mapping. The traced batch-16 run counts a median of 164 evaluator calls per layer, more than its prompt-database insertions (App. C). The counter excludes the initial mapping and the winner's final replay. Controls count valid evaluator calls and are read at 200, an allowance, not an exact per-case match.

Metrics. We report geometric means of baseline EDP / CARTA EDP, where values above 1 favor CARTA. Repeated runs are pooled by per-layer geomean before aggregation over cases, and wins, ties, and losses use a $\pm 1\%$ tie band. Workload aggregates are geomeans over selected layers, not whole-network execution EDP. Layer-bootstrap intervals describe the observed suite, whereas workload-cluster intervals resample the 10 workloads, keeping their layers and accelerator instances together.

5 Evaluation

5.1 A frozen model beats the fair Tuner

CARTA beats per-task policy training with DeepSeek and with pinned open-weight Qwen3-14B (Table 1, Figure 3). Across three independent DeepSeek runs at batch 16, none of which re-ran a case, the fair Tuner has $1.85\times$ CARTA's EDP ([1.55, 2.17] over workloads; App. C).

The Tuner's seed does not rescue it. Seed 42 is the most favorable to CARTA (Figure 3), yet the per-case best of three seeds (960 evaluator calls) still has $1.43\times$ DeepSeek CARTA's EDP.

However, the fair Tuner is a weak incumbent on the main suite, so search is the more demanding comparison (Figure 4). Aligning the objective and attention batch folding lowers canonical batch-16 Tuner/CARTA EDP from $2.15\times$ for the latency-directed Tuner to $1.97\times$ for the fair Tuner. The shared encoding still wrongly shares the second attention operand across the batch, but excluding attention cases moves the headline and control ratios by at most 0.02 (App. C).

Table 1: CARTA and random search beat the fair Tuner. Ratios are fair-Tuner EDP at 320 evaluator calls / comparator EDP (above 1 favors the comparator), pooled over three DeepSeek runs or 3 Qwen repetitions, with layer-bootstrap intervals and $\pm 1\%$ ties. †Canonical rows keep first completed attempts, as 281 of 810 cases were re-run (App. C).

Comparator	Batch	EDP ratio	Interval	Wins/ties/losses
DeepSeek, pooled	16	**1.85**	[1.72, 1.99]	246/6/18
Qwen3-14B, pooled	16	**1.61**	[1.49, 1.74]	213/8/49
Random search, 500 evaluator calls	16	1.66	[1.57, 1.76]	237/7/26
Secondary evidence: canonical sweep, first completed attempts				
DeepSeek, canonical†	1, 16, 64	1.82	[1.74, 1.91]	691/49/70
DeepSeek, canonical†	16	1.97	[1.82, 2.13]	240/10/20
DeepSeek, canonical†	1	1.55	[1.44, 1.69]	211/29/30

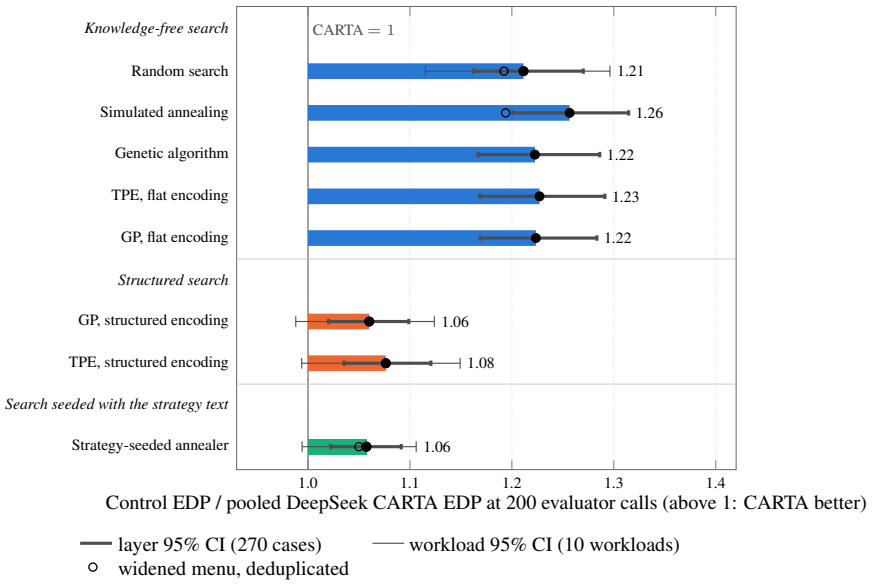

Figure 5: Structural knowledge closes most of DeepSeek CARTA's lead, and workload resampling removes support for the remaining structured-search margins. Bars show control EDP / CARTA EDP (above 1 favors CARTA) with restricted menus at 200 evaluator calls, over 3 seeds and 270 batch-16 cases. CARTA is pooled over three runs, and hollow circles show the widened, deduplicated menu. Thick whiskers (layer-bootstrap 95% intervals) all exclude parity. Thin whiskers over the 10 workloads include parity for structured GP [0.98, 1.12], structured TPE [0.99, 1.15] and the strategy-seeded annealer [0.99, 1.11], but not for random search [1.11, 1.29].

On held-out accelerators, the win depends on the strategy text. Without adaptation, fair-Tuner/DeepSeek EDP is $0.86\times$ on GemminiLike's 30 layers (65 of 90 runs, App. C) and $0.73\times$ on NVDLALike's test layers. After the text port, the test-layer ratios are $1.13\times$ and $1.02\times$ (Sec. 5.4).

5.2 Structural knowledge closes most of the search gap

Figure 5 compares the controls at 200 evaluator calls, above CARTA's traced range plus its initial mapping (Sec. 4), with CARTA pooled over the traced and two September runs (App. C).

Giving search the mapping structure or strategy text closes most of the gap (Figure 5). However, since layers recur across accelerators and share workload structure, workload resampling includes parity for every structured control. The remaining margins therefore show a gain on this suite, not superiority on new workloads.

Hardware and budget. The gain over search concentrates on Simba (Figure 6), while on TensorCore random search ties CARTA and structured GP beats it. The strategy text's value also concentrates there: random-search EDP over the strategy-seeded annealer's is 1.49 on Simba but only

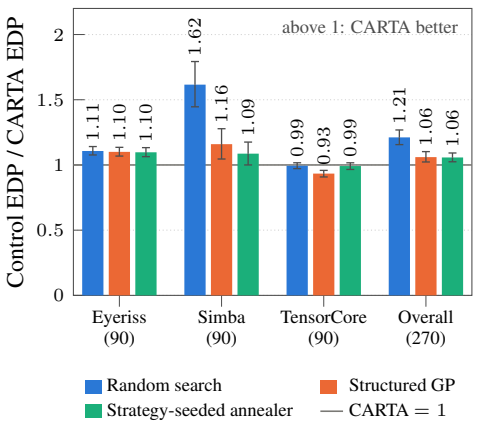

Figure 6: The gain over search concentrates on Simba. Bars show control EDP / pooled DeepSeek CARTA EDP (above 1 favors CARTA) at 200 evaluator calls, with layer-bootstrap 95% whiskers, over 3 control seeds, three DeepSeek runs and 90 batch-16 cases per accelerator. On TensorCore, structured GP beats CARTA.

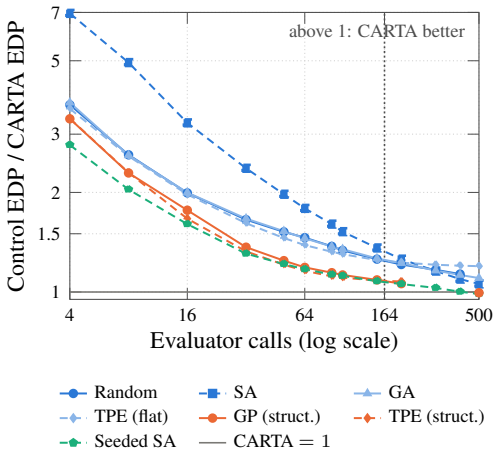

Figure 7: Structured search reaches parity with pooled DeepSeek CARTA at 500 evaluator calls, about 3× its median counted budget. Curves show best-so-far control EDP / CARTA EDP on log axes (lower favors search) over 270 batch-16 cases and 3 control seeds, joining supplied checkpoints. The dotted line marks CARTA's median 164 calls. The isolated GP point is its 500-call aggregate (0.995), and the strategy-seeded annealer reaches 0.986.

1.01 on Eyeriss and 1.00 on TensorCore. Structured search reaches parity at 500 evaluator calls (Figure 7).

5.3 The model determines the increment over structured search

Qwen3-14B keeps the win over the fair Tuner but reverses the comparison with structured search. Its 3 repetitions on the 270 batch-16 cases differ by about ±2%. At 200 evaluator calls, control/Qwen EDP is 1.06 [1.01, 1.11] for random search and 1.10 [1.05, 1.15] for annealing, but only 0.92 [0.89, 0.96] for structured GP and 0.92 [0.89, 0.95] for the strategy-seeded annealer. The same agent implementation therefore does not guarantee an advantage over structured search.

Removing the strategy text exposes the source of DeepSeek's advantage, raising EDP by 1.19× on 17 paired, same-day Simba batch-16 cases. Over its 34 cases, the ablated agent still beats four of the five knowledge-free searches at 200 evaluator calls but ties flat TPE, and the structured controls have 0.83–0.88 times its EDP. Qwen's EDP ratio without versus with strategy text is 0.99 [0.86, 1.11] on 30 Simba layers with 3 repetitions.

5.4 Ports transfer strategy through text

On both ports, strategy text turns DeepSeek's losses to knowledge-free search into wins (Figure 8). GemminiLike and NVDLALike were excluded from the main suite's strategy development, and each uses 30 batch-16 layers split into 10 development and 20 test layers. We transcribed each port's strategy text from the best random-search mappings on the 10 development layers and the unported agent's development failures. This costs 3,000 diagnostic evaluator calls plus expert reading, whereas the Tuner spends 9,600 evaluator calls (320 per layer) on the same 30 layers with no expert. After the first strategy text failed, GemminiLike also spent about 2,500 agent evaluator calls checking the final text on development layers before testing. The resulting text, about 30 lines, measures the artifact's size, not its acquisition cost.

The split holds test layers out of diagnosis and strategy development, not all prior exposure. GemminiLike test layers had been run with the unported agent and the first strategy text before the split was fixed, and NVDLALike had appeared in unrelated runs. Workload families are not held out, so these experiments test adaptation within the benchmark's workloads (App. C).

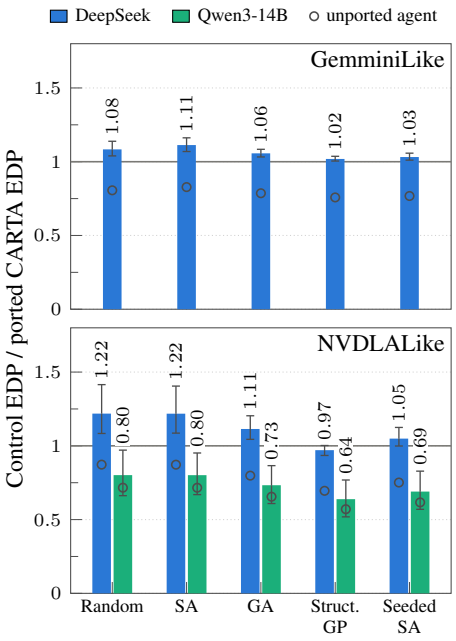

Figure 8: Strategy text turns DeepSeek's losses to knowledge-free search into wins on both ports, while the registered Qwen port still loses. Bars show control EDP / ported CARTA EDP (above 1 favors CARTA) on 20 test layers per accelerator, with controls at 200 evaluator calls and layer-bootstrap 95% whiskers over the 20 layers. Hollow circles: the same control against the unported agent. On NVDLALike, Qwen3-14B is the registered primary backend and DeepSeek the declared secondary. There, the intervals against random search are [1.08, 1.41] for DeepSeek and [0.66, 0.97] for Qwen, and DeepSeek has [0.94, 1.00] against structured GP. The strategy-seeded annealer comes close to DeepSeek, with [1.00, 1.13] on NVDLALike and [1.01, 1.06] on GemminiLike, and beats Qwen at [0.57, 0.83].

Table 2: Evaluation defaults inflate CARTA's apparent advantage in this study. Ratios are control EDP / CARTA EDP (above 1 favors CARTA). The incumbent row uses stored canonical results, the failed-runs row corrects attempt selection, and rows compare distinct protocols that are not additive.

Choice	Protocol comparison	Measured effect
Budget unit	Database insertions → evaluator calls (median 164)	Eight searches: 1.12–1.60 at 88 → 1.06–1.36 at 150/200.
Incumbent and problem	Latency objective, mismatched batch folding → EDP, identical problems	Tuner: 2.05 → 1.85 over 810 cases; unported GemminiLike: 1.76 → 0.86. Problems differed in 18/270 batch-16 cases.
Run selection	Best single run → pool three reproducible runs	Knowledge-free: 1.29–1.34 → 1.21–1.26; structured: 1.12–1.14 → 1.06–1.08.
Failed runs	Re-run cases that look bad → keep first completed attempt	Tuner, 810 cases: 1.85 → 1.82.
Uncertainty unit	Layers → whole workloads (10 clusters)	Structured CIs: exclude → include parity; strategy-seeded annealer 1.06 [0.99, 1.11]. Random stays above parity: [1.11, 1.29].
Control space	Restricted menu → widened menu and deduplication	Random: 1.21 → 1.19; annealing: 1.26 → 1.19; strategy-seeded annealer: 1.06 → 1.05.
Control knowledge	Knowledge-free search → search given the same text	DeepSeek: ∼ 1.2 → 1.05–1.08; Qwen: 0.92 against the strategy-seeded annealer.

GemminiLike: correct spatial fragmentation. On test layers, search has 0.76–0.83 times the unported DeepSeek agent's EDP, and the first strategy text, written from the architecture file alone, worsens this EDP (App. C). Correcting the fragmentation of Sec. 2, the revised text cuts test EDP by 1.34× [1.17, 1.58] relative to the unported agent.

NVDLALike: cap the innermost tile. Both unported agents lose to search (hollow circles in Figure 8), including Qwen3-14B, the backend registered as primary before the port's agent runs, with DeepSeek declared secondary. On transformer GEMMs, the unported agent places $N = 1024$–2048 at Registers (Qwen3-14B in 9 of 9 development runs, DeepSeek in 14 of 18 test runs), while the best development mappings cap it at 4–32. The text reduces DeepSeek's EDP by 1.40× [1.18, 1.68] relative to its unported version. However, random search, annealing and the GA at 500 evaluator calls only tie the seed-42 fair Tuner (0.94–1.01), as does ported DeepSeek at 1.02 [0.94, 1.10]. The registered Qwen3-14B port loses at 0.67–0.73 across Tuner seeds (App. C).

What transfers. The strategy text supplies most of the adaptation, with a small DeepSeek increment: the strategy-seeded annealer comes within $1.05\times$ of DeepSeek on both ports (Figure 8). However, the registered primary prediction on NVDLALike fails, as Qwen improves by $1.12\times$ over its unported version but still loses to every search control (App. C). The text thus transfers a useful strategy, but applying it through an LLM does not ensure a better mapping than search using it directly.

Evaluation choices change the conclusion. Evaluation defaults inflate CARTA's apparent gain (Table 2), and workload resampling changes the support, not the point estimate. Evaluations should therefore count objective calls, align problems and knowledge, disclose run selection, and resample the claimed unit of generalization.

6 Discussion

Where the model adds value. The text carries the gain, the model the increment. For DeepSeek the strategy text is worth $1.19\times$ on Simba, but for Qwen3-14B none is established (0.99 [0.86,1.11] without vs. with). The strategy-seeded annealer recovers most of DeepSeek's benefit and beats Qwen. LLM initialization then search is untested, so repeated LLM calls' value is unisolated.

Evaluator calls and elapsed time. Fewer evaluator calls buy no wall-clock lead, since API latency dominates the traced agent's 279 s per layer. The latency-directed Tuner's trace takes 99.6 s, the fair Tuner's rerun 350 s (median, four-process cap). At equal wall-clock, knowledge-free search at 500 calls trails pooled runs by 1.06–$1.20\times$ (canonical run: 1.12–$1.27\times$). However, random search at 2,000 calls, affordable with a 0.1 s evaluator in CARTA's wall-clock, reaches parity ($1.01\times$ [0.97,1.05], App. C).

Contamination and benchmark exposure. Anonymization leaks no real name in any phase (audited). Full anonymization shows no detected degradation on 30 Simba batch-16 layers (one run each), though anonymized runs use more evaluator calls and keep identifying capacities and fanouts. Pretraining contamination is not ruled out, and the prompts saw seven versions on this benchmark.

7 Related Work

Mapping representations and search. Halide (Ragan-Kelley et al., 2013) and TVM (Chen et al., 2018a) split computation from schedule. AutoTVM (Chen et al., 2018b), FlexTensor (Zheng et al., 2020b), Ansor (Zheng et al., 2020a), and GAMMA (Kao & Krishna, 2020) search schedules per task. CoSA (Huang et al., 2021) casts mapping as a mixed-integer program, Mind Mappings (Hegde et al., 2021) learns a surrogate, and Soter (Wang et al., 2024) trains an RL proposer per task. Timeloop (Parashar et al., 2019), ZigZag (Mei et al., 2021), and MAESTRO (Kwon et al., 2020) model mapping costs. Our baselines, re-implementations including a CoSA-labeled heuristic, show no superiority over native mappers. CARTA keeps Soter's builder and evaluator, swaps the proposer for a frozen LLM and, like search and constrained optimization, trains nothing per task.

LLMs as optimization components. OPRO (Yang et al., 2024) conditions proposals on evaluated solutions, EvoPrompt (Guo et al., 2024) pairs LLM proposals with evolutionary search, and LLAMBO (Liu et al., 2024) uses frozen LLMs as initializers and surrogates. CARTA follows this proposal–evaluation pattern but, unlike the code search of FunSearch (Romera-Paredes et al., 2024) and AlphaEvolve (Novikov et al., 2025), proposes compact decisions a fixed builder completes.

Summary. A frozen LLM given the strategy text beats per-task training, but the text carries the port. Without it the LLM loses on held-out accelerators, and an annealer seeded with it comes within 1.03–$1.06\times$ of DeepSeek, reaching parity at 500 calls. The model sets the increment, small for DeepSeek over structured search, while Qwen3-14B trails it and fails the registered port prediction.

AI USE STATEMENT

This paper studies a frozen large language model (LLM) as an optimizer for tensor mapping, so LLM use is both the object of study and part of our research pipeline; we disclose both roles here.

Research execution (required disclosure). The LLM is the method under evaluation: every CARTA mapping decision reported here is proposed by a frozen, publicly available LLM (primarily DeepSeek-Chat, with GPT-4o-mini, Gemini 2.5 Flash-Lite, and locally served open-weight Qwen models as backend-sensitivity arms), driven by the prompts and search loop of Section 3 and reproduced verbatim in Appendix A. No model was fine-tuned, and no model was given a mapping, target EDP, or solution from any baseline tool on the evaluated layers; the two port strategy texts were hand-written after inspecting the random-search winners on the ten development layers (Appendix C). Every LLM-produced mapping is mechanically validated by a deterministic constraint-satisfying builder and scored by the Timeloop analytical model before any number enters the paper, so no reported result depends on an unverified model output.

Writing and analysis assistance (required disclosure). We used general-purpose AI coding and writing assistants to implement and run the experimental harness, to analyze and tabulate measured results, and to draft and polish this manuscript's prose. The authors checked every claim, table, and figure against the underlying measured artifacts, and the experimental arms were executed through direct API calls rather than through an agentic assistant, so assistant tooling could never observe or influence a measured run.

Not used. We did not use generative AI to generate synthetic datasets, and we make no mathematical claims proved with AI assistance; these categories do not apply to this work.

We have reviewed all AI-assisted work. The authors take full responsibility for the final content of this paper, including all text, claims, code, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work optimizes the scheduling of tensor programs onto hardware accelerators in simulation. It involves no human subjects, no personally identifiable data, and no newly released dataset. All workloads are standard public DNN benchmarks; the three main hardware targets are published academic accelerator models, and the two additional targets (held out of the main suite and of strategy-text development, but used in May-2026 pilot campaigns) are our own Timeloop approximations of published designs, included in the supplementary material. The primary societal consideration is efficiency: better mappings reduce the energy per inference, though efficiency gains can also enable wider deployment. So that the method's limits are not overstated, we report losses and ties alongside wins (Section 5) and dissect a failure case in Appendix C. We declare no conflicts of interest and have read and adhere to the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

Two sources of variation affect our numbers: the builder revision and LLM sampling. The scoring pipeline (smart builder and Timeloop analytical model) is deterministic for a fixed builder revision. Two revisions were used: the earlier one rebuilds all 270 traced winners exactly and served the traced run and, by date, the canonical sweep; the current one served the September runs and all controls. Both are included; Appendix C reports the comparison restricted to one revision. LLM sampling is not deterministic, and CARTA is not seeded. The matched-budget batch-16 headline therefore pools the three later of four independent full-suite DeepSeek runs of the 270 batch-16 cases; none of the three re-ran any case, and the fourth, the March 2026 canonical run and the most favorable, is reported separately. The 810-case comparison against the Tuner comes from the canonical sweep, in which 281 cases were re-run: 261 after a logged API failure (a first attempt with zero evaluator calls) and twenty-four (four of them among the 261) by three outcome-triggered ($> 3\times$) repair passes, three of the twenty-four with no logged preceding attempt. We report that sweep with each case's first completed attempt and give the full attempt ledger in Appendix C.

Backends: DeepSeek-Chat via the public API, accessed March–April 2026 (canonical sweep), May 2026 (traced run) and September 2026 (mechanism ablation, the two pooled full-suite runs, ports and contamination control); GPT-4o-mini via the OpenAI API; Gemini 2.5 Flash-Lite via the Google

API; and Qwen3.5-35B-A3B and Qwen3-14B served locally with vLLM 0.18.0, with Qwen3.5-9B and Qwen3.5-122B-A10B-FP8 served locally for the 12-layer calibration only. No per-case CSV or run log of the September DeepSeek runs or of the Qwen3-14B arms records the served model. The September DeepSeek runs are identified by the API model name, the run dates and the driver source: the drivers request the alias `deepseek-chat`, which serves whatever version the provider deploys on the run date, so the served version is not recorded. The difference between the March canonical run and the pooled September runs is reported in Section 5. All Qwen3-14B arms (the pooled 270-case repetitions, the $0.99\times$ prior null, the $1.41\times$ ablated margin, the first GemminiLike strategy text and the second text's development check, and the NVDLALike arms) are identified by the model id `qwen3-14b` in the driver source, the local vLLM server above, run metadata and directory structure.

The ports of Sec. 5.4 and the contamination control were run with frozen artifacts. The development/test split and the port's strategy text (about 30 lines) were hashed before the test layers were run with the ported agent; the GemminiLike test layers had already been run with the unported agent and the first strategy text before the split was frozen (Appendix C). The predictions were recorded, with timestamps, in plain-text files that are not hashed; the file for the contamination control records an interim look and the arm added after it. It also records two withdrawn implementations of the anonymization. The first left the short level-name forms of the strategy text unreplaced, and its interim fully-anonymized/named ratio ($0.80\times$ on 21 layers) is not used. The second was withdrawn after names were found in the refine and polish phases ($0.87\times$ and $1.01\times$). All three arms were re-run with the third implementation, which is the one reported (Appendix C). All of these files, the per-case result files, the driver scripts and the falsified fanout prediction are included in the supplementary material.

Section 4 specifies the three accelerator models, the 90-layer workload suite, the batch settings, and the fixed LLM-call budget; Appendix A reproduces the system prompt and an explore-phase context prompt verbatim, and Appendix G gives the search algorithm and the per-phase prompt schema. Evaluation uses the open-source Timeloop analytical model with the published accelerator configurations of Appendix I, so a third party can re-score any mapping we report. Because LLM sampling is stochastic, we report the observed run-to-run variation and, in Section 5.3 and Appendix C, independent re-runs of representative layers that reproduce the median layer within 2% but not every large first-layer win. Per-call token counts and costs are tabulated in Appendix D. Source code, prompts, and the full per-case result tables are included in the supplementary material.

REFERENCES

Baidu Research. DeepBench: Benchmarking deep learning operations. `https://github.com/baidu-research/DeepBench`, 2017.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *NeurIPS*, 2020.

Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, Carlos Guestrin, and Arvind Krishnamurthy. TVM: An automated end-to-end optimizing compiler for deep learning. In *OSDI*, 2018a.

Tianqi Chen, Lianmin Zheng, Eddie Yan, Ziheng Jiang, Thierry Moreau, Luis Ceze, Carlos Guestrin, and Arvind Krishnamurthy. Learning to optimize tensor programs. In *NeurIPS*, 2018b.

Yu-Hsin Chen, Joel Emer, and Vivienne Sze. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. In *ISCA*, 2016.

Chris Cummins, Volker Seeker, Dejan Grubisic, Baptiste Roziere, Jonas Gehring, Gabriel Synnaeve, and Hugh Leather. Large language model compiler: Foundation models of compiler optimization. *arXiv preprint arXiv:2407.02524*, 2024.

Shail Dave, Youngbin Kim, Sasikanth Avancha, Kyoungwoo Lee, and Aviral Shrivastava. dmazerunner: Executing perfectly nested loops on dataflow accelerators. *ACM Transactions on Embedded Computing Systems*, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

Yonggan Fu et al. Gpt4aigchip: Towards next-generation ai accelerator design automation via large language models. *arXiv:2309.10730*, 2023.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2309.08532.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Kartik Hegde, Po-An Tsai, Sitao Huang, Vikas Chandra, Angshuman Parashar, and Christopher W. Fletcher. Mind mappings: enabling efficient algorithm-accelerator mapping space search. In *ASPLOS*, 2021.

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

Qijing Huang, Minwoo Kang, Grace Dinh, Thomas Norell, Aravind Kalaiah, James Demmel, John Wawrzynek, and Yakun Sophia Shao. Cosa: Scheduling by constrained optimization for spatial accelerators. In *ISCA*, 2021.

Sheng-Chun Kao and Tushar Krishna. Gamma: Automating the hw mapping of dnn models on accelerators via genetic algorithm. In *ICCAD*, 2020.

Sheng-Chun Kao, Suvinay Subramanian, Gaurav Agrawal, Amir Yazdanbakhsh, and Tushar Krishna. FLAT: An optimized dataflow for mitigating attention bottlenecks. In *ASPLOS*, 2023.

Hyoukjun Kwon, Prasanth Chatarasi, Michael Pellauer, Angshuman Parashar, Vivek Sarkar, and Tushar Krishna. Understanding reuse, performance, and hardware cost of DNN dataflows: A data-centric approach using MAESTRO. *IEEE Micro*, 2020.

Jianling Li, Shangzhan Li, Zhenye Gao, Qi Shi, Yuxuan Li, et al. TritonBench: Benchmarking large language model capabilities for generating Triton operators. *arXiv preprint arXiv:2502.14752*, 2025.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with AlphaCode. *Science*, 378(6624):1092–1097, 2022a.

Zhaoying Li, Dan Wu, Dhananjaya Wijerathne, and Tulika Mitra. LISA: Graph neural network based portable mapping on spatial accelerators. In *HPCA*, 2022b.

Mengfan Liu, Da Zheng, Junwei Su, and Chuan Wu. Step-tp: A grounded, step-level dataset with chain-of-thought reasoning for llm-guided tensor program optimization. *arXiv preprint arXiv:2605.25954*, 2026.

Tennison Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. Large language models to enhance bayesian optimization. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2402.03921.

Linyan Mei, Pouya Houshmand, Vikram Jain, Sebastian Giraldo, and Marian Verhelst. Zigzag: Enlarging joint architecture-mapping design space exploration for dnn accelerators. In *IEEE Transactions on Computers*, volume 70, pp. 1160–1174, 2021.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, et al. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

NVIDIA. Tensor core. *NVIDIA GPU Architecture White Paper*, 2018. URL `https://www.nvidia.com/data-center/tensor-cores`.

Anne Ouyang, Simon Guo, Simran Arora, Alex L. Zhang, William Hu, Christopher Ré, and Azalia Mirhoseini. KernelBench: Can LLMs write efficient GPU kernels? *arXiv preprint arXiv:2502.10517*, 2025.

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 *ISPASS*, 2019.

Jonathan Ragan-Kelley, Connelly Barnes, Andrew Adams, Sylvain Paris, Frédo Durand, and Saman Amarasinghe. Halide: A language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines. In *PLDI*, 2013.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Kai Olav Ellefsen, et al. Mathematical discoveries from program search with large language models. *Nature*, 625:468–475, 2024.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

Enrico Russo et al. MEDEA: A multi-objective evolutionary approach to DNN hardware mapping. In *DATE*, 2022.

Yakun Sophia Shao, Jason Clemons, Rangharajan Venkatesan, Brian Zimmer, Matthew Fojtik, Nan Jiang, Ben Keller, Alicia Klinefelter, Nathaniel Pinckney, Priyanka Raina, et al. Simba: Scaling deep-learning inference with multi-chip-module-based architecture. In *MICRO*, 2019.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Arne Symons, Linyan Mei, and Marian Verhelst. LOMA: Fast auto-scheduling on DNN accelerators through loop-order-based memory allocation. In *IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

Fuyu Wang, Minghua Shen, Yufei Ding, and Nong Xiao. Soter: Analytical tensor-architecture modeling and automatic tensor program tuning for spatial accelerators. In *ISCA*, 2024.

Jianghui Wang, Vinay Joshi, Saptarshi Majumder, Xu Chao, Bin Ding, et al. GEAK: Introducing Triton kernel AI agent and evaluation benchmarks. *arXiv preprint arXiv:2507.23194*, 2025.

Qingcheng Xiao et al. Hasco: Towards agile hardware and software co-design for tensor computation. In *ISCA*, 2021.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2309.03409.

Xuan Yang, Mingyu Gao, Qiaoyi Liu, Jeff Setter, Jing Pu, Ankita Nayak, Steven Bell, Kaidi Cao, Heonjae Ha, Priyanka Raina, Christos Kozyrakis, and Mark Horowitz. Interstellar: Using halide's scheduling language to analyze dnn accelerators. In *ASPLOS*, 2020.

Lianmin Zheng, Chengfan Jia, Minmin Sun, Zhao Wu, Cody Hao Yu, Ameer Haj-Ali, Yida Wang, Jun Yang, Danyang Zhuo, Koushik Sen, Joseph E. Gonzalez, and Ion Stoica. Ansor: Generating high-performance tensor programs for deep learning. In *OSDI*, 2020a.

Size Zheng, Yun Liang, Shuo Wang, Renze Chen, and Kaiwen Sheng. Flextensor: An automatic schedule exploration and optimization framework for tensor computation on heterogeneous system. In *ASPLOS*, 2020b.

Size Zheng, Renze Chen, Pengcheng Duan, Xiuhong Wang, and Yun Liang. Amos: Enabling automatic mapping for tensor computations on spatial accelerators with hardware abstraction. In *ISCA*, 2022.

A Example Explore-Phase Prompt

The simplest way to show that the LLM's input carries no oracle from prior mappers is to print it. Below, unedited, are the system prompt and the Phase-1 (Explore) context prompt that the agent builds for Simba running ResNet-50 layer 10 at batch 16. The released code's prompt builders generated both; for display we set non-ASCII arrows, multiplication signs and one em dash in ASCII and wrapped six long lines (four spatial-menu lines, one data-reuse line and the Simba hierarchy line), marking continuations "–>". The system prompt depends on the accelerator (it carries that accelerator's strategy text) and on the layer type (GEMM, depthwise, pointwise or standard convolution); the context prompt is rebuilt for every layer. Two display artifacts of the agent as run remain uncorrected: the level list is headed "inner → outer" but printed outermost first, and the spatial menus repeat some single-dimension entries.

```
=== SYSTEM ===
You are a DNN accelerator mapping expert optimizing EDP = energy x cycles.

## How the mapping builder works
Your JSON decisions are converted to a mapping:

1. **Spatial allocation** (`{Level}_spatial_{Dim}: value`):
   Product of spatial values MUST equal the PE count for that level.
   Spatial dims are parallelized -- data along non-spatial dims is multicast.
   - K-spatial -> inputs multicast (shared across K PEs)
   - N-spatial -> weights multicast (shared across N PEs)
   - C-spatial -> partial sums need accumulation (extra traffic)

2. **Priority ordering** (`{Level}_priority: [dim_list]`):
   The builder fills each buffer GREEDILY in priority order:
   - 1st dim gets the LARGEST possible tile (fills most of the buffer)
   - 2nd dim gets next largest from remaining capacity
   - Last dims may get tile=1 (must be tiled at outer levels -> more DRAM traffic)
   Priority is THE most important decision. It determines data reuse.

3. **Tile caps** (`{Level}_{Dim}: value`):
   Limits a dim's tile at that level. Remaining factor goes to outer levels.
   Useful for balancing tile distribution across levels.

## Data reuse patterns
- **Weights** depend on [H,C,K,R,S]: reused across N,P,Q iterations
  -> At Weights-keeping levels: put C,K,R,S first in priority
  -> P,Q,N are FREE (don't affect weight tile size)
- **Inputs** depend on [N,H,C,P,Q]: reused across K iterations
  -> At Inputs-keeping levels: put C,P,Q first in priority
- **Outputs** depend on [N,H,K,P,Q]: reused across C,R,S iterations
  -> At Outputs-keeping levels: K,P,Q,N first. C early CAN help when C is spatially
  -> split.

## Energy costs
DRAM: ~200pJ/word. On-chip: ~1-5pJ/word. DRAM is 100x more expensive!
-> Maximize data reuse in on-chip buffers to avoid DRAM traffic.
-> The OUTERMOST temporal loops determine what streams from DRAM.

## Pointwise conv (1x1, R=S=1)
Similar to GEMM but with spatial dims P,Q.
- P,Q spatial at inner levels -> output locality
- CxK spatial at outer levels -> weight reuse across batches
- At Outputs-only: K,P,Q first. Cap inner C small for outer weight reuse.

## Cap strategy for inner levels
At Weights-keeping inner levels, P,Q,N don't affect weight tile size.
But they DO affect output tile size at Outputs-keeping levels above.
Capping inner P=2,Q=2 instead of P=1,Q=1 can give 10x better EDP
because outer levels get better output tile distribution.
Try different cap values: 1, 2, 4, etc. for free dims.
```

```
## Recommended Simba strategies (try ALL of these):
Simba has: Registers(Weights,1B) -> AccBuf(Outputs,sp=64) -> WeightBuf -> InputBuf ->
  -> GlobalBuf(I+O,sp=16) -> DRAM

Strategy A - Multi-dim spatial + inner caps:
  AccBuf spatial: C=8,K=8 or K=4,C=4,N=4 or P=4,Q=4,K=4
  GlobalBuf spatial: C=16 or H=16 or N=16
  Registers caps: P=2,Q=2 (CRITICAL for conv! Lets outer levels tile P,Q)
  Priorities: AccBuf=[K,P,Q,C,N,R,S], Registers=[C,K,R,S,P,Q,N]

Strategy B - K-heavy output-stationary:
  AccBuf spatial: K=64 (all outputs parallel)
  GlobalBuf spatial: C=16 or N=16
  Priorities: all levels=[K,C,P,Q,N,R,S]

Strategy C - N-spatial weight multicast:
  AccBuf spatial: N=64 or N=8,K=8
  GlobalBuf spatial: N=16
  Priorities: AccBuf=[K,N,P,Q,C,R,S], Weight/InputBuf=[C,K,R,S,P,Q,N]

Strategy D - Mixed per-level priorities:
  AccBuf: K first (output-stationary), WeightBuf: C,K first (weight reuse)
  Try with various spatial + Registers caps P=1..4, Q=1..4, N=1..4


=== CONTEXT (phase 1) ===
# Architecture: Simba (1024 PEs)
Levels (inner -> outer):
  GlobalBuffer: size=65536, spatial=16 PEs, stores=Inputs+Outputs
  InputBuffer: size=8192, stores=Inputs
  WeightBuffer: size=32768, stores=Weights
  AccumulationBuffer: size=3072, spatial=64 PEs, stores=Outputs
  Registers: size=1, stores=Weights

# Problem: R=1, S=1, P=56, Q=56, C=64, K=256, N=16, H=1 (Conv)
Active dims: P=56, Q=56, C=64, K=256, N=16

## GlobalBuffer spatial options (product MUST = 16):
  1-dim: {C=16}, {K=16}, {N=16}, {K=16}, {N=16}, {N=16}
  2-dim: {C=2, K=8}, {C=4, K=4}, {C=8, K=2}, {C=2, N=8}, {C=4, N=4}, {C=8, N=2},
    -> {C=2, P=8},
    -> {C=4, P=4}, {C=8, P=2}, {C=2, Q=8}, {C=4, Q=4}, {C=8, Q=2}, {K=2, N=8}, {K=4,
    -> N=4}, {K=8, N=2}
    -> (+15 more)
  3-dim: {N=2, P=2, Q=4}, {N=2, P=4, Q=2}, {N=4, P=2, Q=2}, {K=2, P=2, Q=4}, {K=2,
    -> P=4, Q=2},
    -> {K=2, N=2, Q=4}, {K=2, N=2, P=2, Q=2}, {K=2, N=2, P=4}, {K=2, N=4, Q=2}, {K=2,
    -> N=4, P=2},
    -> {K=4, P=2, Q=2}, {K=4, N=2, Q=2}, {K=4, N=2, P=2}, {C=2, P=2, Q=4}, {C=2, P=4,
    -> Q=2} (+20 more)
  (Or propose any combo where product = 16 and each value divides its dim)

## AccumulationBuffer spatial options (product MUST = 64):
  1-dim: {C=64}, {K=64}, {K=64}
  2-dim: {C=2, K=32}, {C=4, K=16}, {C=8, K=8}, {C=16, K=4}, {C=32, K=2}, {C=4,
    -> N=16}, {C=8,
    -> N=8}, {C=16, N=4}, {C=32, N=2}, {C=8, P=8}, {C=16, P=4}, {C=32, P=2}, {C=8,
    -> Q=8}, {C=16, Q=4},
    -> {C=32, Q=2} (+15 more)
  3-dim: {N=2, P=4, Q=8}, {N=2, P=8, Q=4}, {N=4, P=2, Q=8}, {N=4, P=4, Q=4}, {N=4,
    -> P=8, Q=2},
    -> {N=8, P=2, Q=4}, {N=8, P=4, Q=2}, {N=16, P=2, Q=2}, {K=2, P=4, Q=8}, {K=2, P=8,
    -> Q=4}, {K=2,
    -> N=2, P=2, Q=8}, {K=2, N=2, P=4, Q=4}, {K=2, N=2, P=8, Q=2}, {K=2, N=4, Q=8},
    -> {K=2, N=4, P=2,
    -> Q=4} (+128 more)
  (Or propose any combo where product = 64 and each value divides its dim)

## Valid cap values (divisors of each dim):
  P=56: [1, 2, 4, 7, 8, 14, 28, 56]
  Q=56: [1, 2, 4, 7, 8, 14, 28, 56]
  C=64: [1, 2, 4, 8, 16, 32, 64]
  K=256: [1, 2, 4, 8, 16, 32, 64, 128, 256]
  N=16: [1, 2, 4, 8, 16]

## Decision keys to output:
  GlobalBuffer_priority: [list of P, Q, C, K, N] (stores: Inputs+Outputs)
  InputBuffer_priority: [list of P, Q, C, K, N] (stores: Inputs)
  WeightBuffer_priority: [list of P, Q, C, K, N] (stores: Weights)
  AccumulationBuffer_priority: [list of P, Q, C, K, N] (stores: Outputs)
```

```
Registers_priority: [list of P, Q, C, K, N] (stores: Weights)
GlobalBuffer_spatial_{dim}: value (product = 16)
AccumulationBuffer_spatial_{dim}: value (product = 64)
Optional caps: {Level}_{Dim}: value (must be a divisor of dim)
```

The prompt contains only five things: (i) the architecture hierarchy parsed from `arch.yaml`, (ii) the problem dimensions parsed from `problem.yaml`, (iii) the spatial factorizations enumerated mechanically from the architecture constraints, (iv) the accelerator's hand-written strategy text, and (v) the schema the LLM must emit. No reference EDP, no Soter mapping and no oracle target appear. Each Explore call appends the five best PromptDatabase entries so far (EDP, cycles, utilization, their decisions and per-level energy shares) and a task block for that call (a one-line scripted strategy hint in the first twelve calls); at the first call the history holds only the builder's initial mapping, whose EDP is the agent's own baseline. Refine and Polish add the parent configuration and its energy breakdown. In Phase 1 the initial-mapping EDP is the only reference EDP the LLM sees; no baseline mapper's EDP enters the prompt.

B Benchmark Layer Selection

The 90-layer suite is a fixed selection from the ten workload specifications shipped with Soter (Wang et al., 2024); Table 3 lists the layers by their position within each workload's specification file. The same 90 layers are run on every accelerator at batch 1, 16 and 64: 270 cases per batch size, 810 in total. The selection was fixed once, before any agent or control ran on this suite, and never changed. It was preceded by a 30-layer pilot set (three evenly spaced layers per workload) on which the agent and the March-2026 random and annealing controls were run; of its 30 layers, 23 are in the 90-layer suite and 7 (DenseNet-161 53, 106; ResNet-50 18, 36; ResNeXt-50 18, 36; UNet 7) are not. App. H tabulates the per-task training and evaluator-call budgets of each paradigm.

Table 3: Benchmark layer selection (90 layers from 10 workloads). Indices are positions within each workload specification as shipped with Soter; the full per-layer tensor dimensions are reproduced in the supplementary material.

Workload	Layers	Layer indices
ResNet-50	9	0, 5, 10, 13, 27, 35, 40, 45, 53
ResNeXt-50 (32×4d)	9	0, 5, 10, 13, 27, 35, 40, 45, 53
DenseNet-161	9	0, 32, 40, 64, 80, 96, 120, 128, 160
VGG-16	12	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
MobileNet	12	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
UNet	12	0, 2, 4, 6, 8, 9, 10, 12, 14, 16, 18, 19
GPT-1	6	0, 1, 2, 3, 4, 5
GPT-3	6	0, 1, 2, 3, 4, 5
BERT-Large	6	0, 1, 2, 3, 4, 5
DeepBench	9	0, 1, 2, 3, 4, 5, 6, 7, 8
Total	**90**	

Configuration and budget accounting (from Sec. 4). Sampling temperature starts at 0.7, 0.6, and 0.4 in the three phases, rising 0.03 per stagnant call in the first two (capped at 1.2 in exploration) and 0.02 per call in polishing. Pinned open-weight Qwen3-14B runs locally via vLLM, 3 repetitions on the 270 batch-16 cases. Agent repetitions are unseeded (no sampling seed reaches the API or local server); controls use explicit seeds.

C Additional Evaluation Results

Main-text details. The following sentences accompany Sec. 4 and Sec. 5. Soter-suite baselines labeled AutoTVM (Chen et al., 2018b), FlexTensor (Zheng et al., 2020b), Ansor (Zheng et al., 2020a), and CoSA (Huang et al., 2021) are re-implementations inside Soter's abstraction, and the CoSA-labeled one uses a reuse-score heuristic and hill climbing, not CoSA's mixed-integer formulation. The two September runs alone, which share the controls' builder revision and the Tuner's

power-of-two spatial factors, give the same ratio at [1.71, 1.99]. The three runs reproduce one another, although the canonical run has $1.06\times$ lower EDP than the pool overall and $1.27\times$ lower on Simba. However, excluding it is an outcome-dependent choice, and the pool describes these runs, not uncertainty over future served models. Widening the controls' menus and deduplicating leaves the knowledge-free margin (hollow circles, App. C). With pinned Qwen3-14B over 3 repetitions, unported EDP / EDP with this text is 0.85 [0.79, 0.91] on all 30 layers and 0.90 [0.83, 0.97] on the 20 test layers. The registered prediction placed the strategy-seeded annealer within $1.05\times$ of the ported agent, yet Qwen has $1.45\times$ the annealer's EDP.

Table 4: Which runs and baselines each result uses. Builder entries describe CARTA; every search control uses the current builder, and the fair Tuner uses its own serializer. "n/a" marks information not established for that result family. Controls count valid evaluator calls; CARTA's counter excludes its initial-mapping evaluation (Sec. 4).

Result family	CARTA runs used	Tuner	Builder revision	Budget unit	Re-run policy
Headline Tuner comparison	Pooled traced May + two September DeepSeek runs	Fair	`2714734` + current	Evaluator calls	None
Canonical 810-case sweep	March DeepSeek; first completed attempts	Fair	`2714734` (by date)	40 LLM calls; evaluator count n/a	API-failure and outcome-triggered re-runs; first completed attempt kept
Controls at 200/500	Pooled DeepSeek in main text; stored canonical in App. C tables	Fair in Tuner rows; none otherwise	`2714734` + current (pool); `2714734` by date (canonical)	Evaluator calls	None (pool); stored canonical keeps the last re-run attempt
Widened-menu controls	Pooled DeepSeek	None	`2714734` + current	Evaluator calls	None
Qwen3-14B repetitions	Pooled 3 repetitions, batch 16	Fair	Current	Evaluator calls	85 seed-2 duplicates from a second driver; primary kept
Ports	GemminiLike: DeepSeek (first strategy text and fanout sweep: Qwen3-14B); NVDLALike: Qwen primary, DeepSeek secondary	Fair; latency-directed for correction	Current	Evaluator calls	n/a; GemminiLike unported DeepSeek arm stopped at 65 of 90 runs (API balance)
Contamination control	Same-day DeepSeek, one run per arm	None	Current	Evaluator calls	n/a
Backend/budget single runs (App. J)	Independent single runs; pooled reference labelled	Fair	n/a	LLM calls; budget runs lack evaluator counts	n/a; budget runs used repair scripts
Legacy per-operator figures (Figs. 9, 10, 11)	Stored canonical DeepSeek	Latency-directed	`2714734` (by date)	40 LLM calls; evaluator count n/a	Stored canonical keeps the last re-run attempt
Tables 7, 6	Stored canonical DeepSeek	Fair and latency-directed	`2714734` (by date)	40 LLM calls; evaluator count n/a	Stored canonical keeps the last re-run attempt

Results labelled "latency-directed" are retained for the legacy per-operator figures, the baseline-reproduction tables of Sec. C.2, and to illustrate the corrections. These legacy results, and every result labelled "canonical", are superseded where they differ by the main-text numbers: the fair Tuner, the pooled DeepSeek runs, and parity with structured search at 500 evaluator calls (Sec. 5).

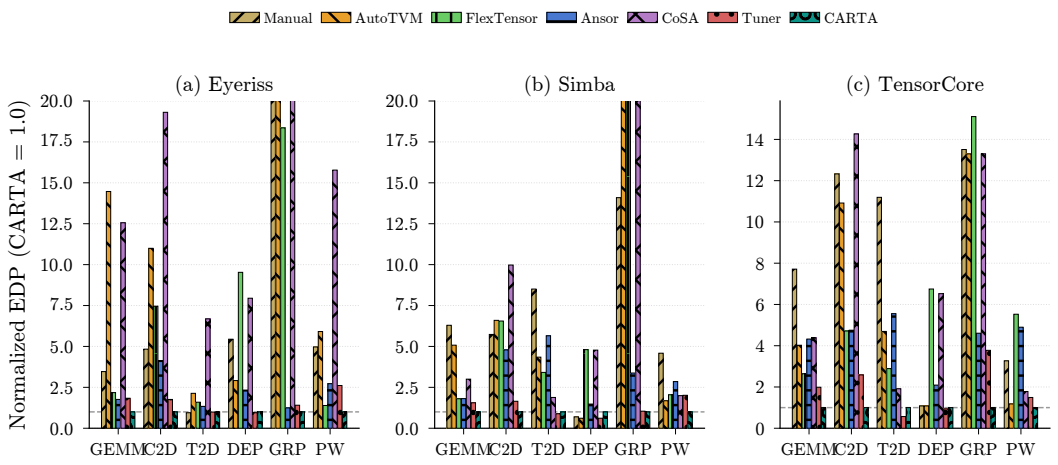

Figure 9: Per-operator EDP speedup at batch size 1, stored canonical DeepSeek results (last re-run attempt) vs the latency-directed Tuner, across three accelerators (Eyeriss, Simba, TensorCore), broken down by six operator types: C2D (standard convolution), T2D (transposed convolution), GEMM (matrix multiply), DEP (depthwise convolution), GRP (group convolution), and PW (pointwise convolution). Values > 1 indicate CARTA wins. At batch 1, DEP and T2D are net losses on all three accelerators (geomean 0.57–$0.99\times$); the largest gains are on PW (Eyeriss, Simba) and on C2D and the single GRP layer (TensorCore).

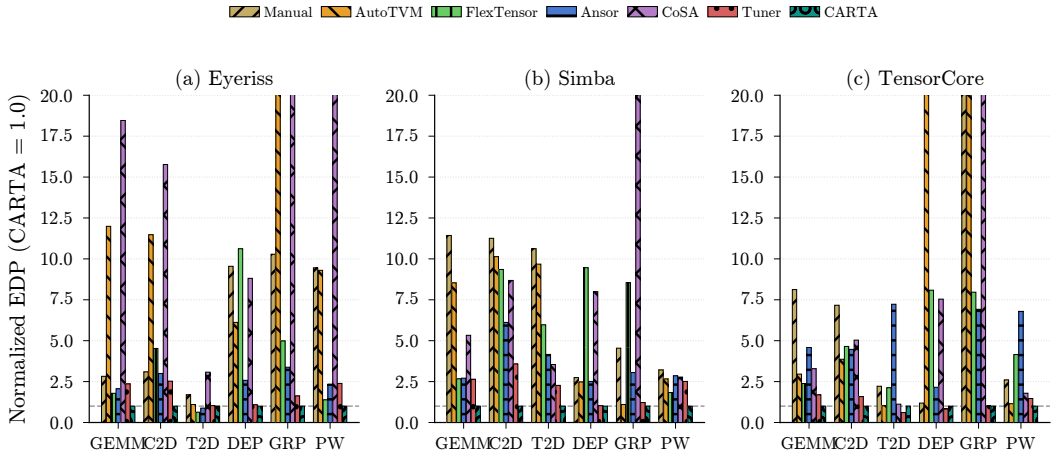

Figure 10: Per-operator EDP speedup at batch size 16, stored canonical DeepSeek results (last re-run attempt) vs the latency-directed Tuner, across six operator types. The margin is larger than at batch 1 on Eyeriss and Simba (up to $18.9\times$ on individual Simba layers) and smaller on TensorCore, where T2D ($0.61\times$) and DEP ($0.83\times$) remain net losses; the strongest gains are on C2D (Simba $3.58\times$).

Because the canonical sweep re-ran cases after outcome-triggered checks, first-completed-attempt accounting applies only where stated ("Re-runs in the canonical sweep" below).

C.1 Per-Operator Analysis

This subsection, including its loss counts, compares the stored canonical DeepSeek results (last re-run attempt) with the latency-directed Tuner; the fair Tuner accounting is in Sec. C.4. Figures 9 and 10 show per-operator-type EDP speedups at batch sizes 1 and 16 across the three accelerators. The suite has six operator types: standard convolution (C2D; 45 layers per accelerator), transposed convolution (T2D; 3), general matrix multiplication (GEMM; 29), depthwise convolution (DEP; 4), group convolution (GRP; 1), and pointwise convolution (PW; 8).

Table 5: Latency-directed Tuner geomean EDP speedup over baselines (Simba). "Paper" column shows the range reported in (Wang et al., 2024). Our reproduction is of the same order; cells deviating from the reported value by more than $1.4\times$ are FlexTensor and Ansor at $B{=}16$ and $B{=}64$ and AutoTVM at $B{=}64$ (below) and CoSA at $B{=}1$ (above).

Baseline	b=1	b=16	b=64	Paper (Wang et al., 2024)
Manual	3.5×	3.2×	2.9×	2.7–3.4×
AutoTVM	3.2×	2.7×	2.0×	3.1×
FlexTensor	2.5×	1.8×	1.3×	2.7×
Ansor	2.1×	1.4×	1.1×	2.6×
CoSA	3.6×	2.3×	2.2×	2.4×

Per-operator geomeans. The geomean ratio (latency-directed Tuner EDP / stored CARTA EDP; Eyeriss/Simba/TensorCore) by operator type is, at batch 1: C2D $1.76/1.65/2.59$, GEMM $1.83/1.55/1.99$, PW $2.61/2.01/1.49$, GRP $1.41/1.04/3.78$ (one layer), DEP $0.95/0.61/0.94$, and T2D $0.99/0.89/0.57$, so DEP and T2D are net losses on all three accelerators. At batch 16: C2D $2.53/3.58/1.58$, GEMM $2.36/2.65/1.69$, PW $2.39/2.51/1.47$, GRP $1.62/1.22/1.01$, DEP $1.09/1.03/0.83$, and T2D $1.03/2.27/0.61$; the net losses are DEP and T2D on TensorCore. At batch 64: C2D $2.85/4.19/1.70$, GEMM $2.26/2.38/1.61$, PW $2.47/2.52/1.53$, GRP $1.96/1.65/1.19$, DEP $1.07/0.95/0.85$, and T2D $1.24/1.17/0.95$; the net losses are DEP on Simba and TensorCore and T2D on TensorCore.

Per-layer spread and losses. Over the 810 cases CARTA wins 735, ties 24, and loses 51 ($\pm 1\%$ band); the median loss is $1.29\times$ in the Tuner's favour. The per-layer ratio ranges from $0.10\times$ to $42.8\times$ (TensorCore ResNet-50 layer 45 at batch 1); on Simba at batch 16 the maximum is $18.9\times$ (DenseNet-161 layer 0), with four layers above $10\times$. The worst loss is Eyeriss UNet layer 10 at batch 16, $10.3\times$ against the stored Tuner result; the same layer is $4.5\times$ against the Tuner re-run used for Table 19. The 51 losses fall on C2D (15 of 405 cases), T2D (14 of 27), DEP (12 of 36), and GEMM (10 of 261); PW (72 cases) and GRP (9) have none. T2D (Ronneberger et al., 2015) is the most variable type (13 wins, 0 ties, 14 losses; 0.10–$4.94\times$). The DEP (Howard et al., 2017) losses are spread over all three accelerators and batch sizes (TensorCore 7, Simba 3, Eyeriss 2; the largest are Simba batch 1 at $3.60\times$ and $1.97\times$), and DEP as a whole is 13 wins, 11 ties, 12 losses. GEMM ratios span 0.68–$31.4\times$ (largest on TensorCore DeepBench layer 8 at batch 1); on the transformer workloads (Vaswani et al., 2017) (BERT (Devlin et al., 2019), GPT (Brown et al., 2020)) they span 0.68–$10.4\times$ with 8 losses of 162, the largest gain on Eyeriss GPT-3 layer 1. The single group-convolution layer (ResNeXt-50 layer 35) ranges 1.01–$3.78\times$ over its nine (accelerator, batch) cells (geomean $1.52\times$); PW layers range 0.99–$5.54\times$ (geomean $2.06\times$; 67 wins, 5 ties, 0 losses); C2D layers range 0.54–$42.8\times$ (387 wins, 3 ties, 15 losses).

Per-accelerator patterns. At batch 16 and 64 the margin is largest on Simba and smallest on TensorCore (latency-directed Tuner/CARTA $2.29/2.90/1.50$ at batch 16 and $2.42/2.96/1.57$ at batch 64, Eyeriss/Simba/TensorCore; fair Tuner $2.19/2.68/1.29$ and $2.21/2.78/1.36$); at batch 1 the order reverses (largest on TensorCore, smallest on Simba: $1.76/1.54/2.06$; fair Tuner $1.57/1.46/1.71$), a pattern we observe but do not explain.

C.2 Comparison with All Baselines

Table 5 checks our baseline reproduction: it compares the latency-directed Tuner's advantage over five traditional baselines on Simba with the ranges reported in the original Soter paper (Wang et al., 2024). The reproduction is of the same order but not exact: six of the fifteen cells deviate from the reported value by more than $1.4\times$ (FlexTensor and Ansor at $B{=}16$ and $B{=}64$ and AutoTVM at $B{=}64$ below it, up to $2.4\times$ for Ansor at $B{=}64$; CoSA at $B{=}1$ above it, $1.5\times$). This suffices because every CARTA comparison is measured directly per layer rather than derived from these ratios.

Every baseline row of Table 7 is measured *directly*, layer by layer, against CARTA on the same 90 layers per configuration, not derived from the Tuner ratios above; Table 6 gives the corresponding win/tie/loss counts. These margins mostly measure search-space restriction rather than search quality: the re-implemented baselines search template-restricted spaces, which is the main reason for their gap to CARTA, and the Tuner itself is beaten by unconstrained random search at 500 evaluator calls, so they should be read together with the unconstrained-search controls (Figure 5 and below).

Table 6: Direct per-layer comparison of CARTA (stored canonical DeepSeek results, last re-run attempt; see "Re-runs in the canonical sweep") against each re-implemented baseline, summed over the nine (accelerator, batch) configurations of Table 7 (810 cases each). Geomean range spans the nine configurations.

Baseline	W	T	L	Non-loss	Geomean range
AutoTVM	717	37	56	93%	3.3×–10.3×
FlexTensor	740	15	55	93%	2.7×–5.3×
Ansor	784	17	9	99%	2.3×–4.9×
CoSA	785	13	12	99%	3.9×–17.8×
Manual	770	20	20	98%	3.5×–9.4×

Table 7: Per-(accelerator, batch) comparison of CARTA (stored canonical DeepSeek results, last re-run attempt; see "Re-runs in the canonical sweep") against both Tuners and the template-constrained re-implementations; the pooled and first-completed-attempt Tuner ratios are in Table 1. The first Tuner row is the fair Tuner (EDP objective, CARTA's exact problem specification, 320 evaluator calls); the second is the latency-directed Tuner, which optimized latency and folded batch into H for attention layers; the re-implementation rows were measured under the latency-directed convention. Geomean EDP normalized to CARTA (=1.0×); higher is worse. Every row is measured directly per layer against CARTA on the same 90 layers (per-configuration W/T/L in App. C). Rows below the Tuner are template-constrained re-implementations inside Soter's mapspace (Sec. 4); the unconstrained controls are in Figure 5.

	Batch = 1			Batch = 16			Batch = 64		
Method	**Eyeriss**	**Simba**	**TC**	**Eyeriss**	**Simba**	**TC**	**Eyeriss**	**Simba**	**TC**
CARTA	**1.0×**	**1.0×**	**1.0×**	**1.0×**	**1.0×**	**1.0×**	**1.0×**	**1.0×**	**1.0×**
Fair Tuner	1.6×	1.5×	1.7×	2.2×	2.7×	1.3×	2.2×	2.8×	1.4×
Latency-directed Tuner	1.8×	1.5×	2.1×	2.3×	2.9×	1.5×	2.4×	3.0×	1.6×
AutoTVM	10.2×	4.9×	5.7×	10.3×	7.8×	3.3×	10.1×	6.1×	3.8×
FlexTensor	4.2×	3.8×	4.0×	2.9×	5.3×	3.7×	2.7×	3.9×	4.5×
Ansor	2.8×	3.2×	4.5×	2.5×	4.1×	4.6×	2.3×	3.2×	4.9×
CoSA	15.4×	5.5×	7.3×	16.2×	6.6×	3.9×	17.8×	6.4×	4.0×
Manual	4.2×	5.4×	8.4×	3.5×	9.4×	6.1×	3.8×	8.6×	5.9×

Batch scaling. Against the latency-directed Tuner, CARTA's geomean margin grows with batch size on Eyeriss and Simba ($1.76 \to 2.29 \to 2.42\times$ and $1.54 \to 2.90 \to 2.96\times$ at batch $1 \to 16 \to 64$; against the fair Tuner $1.57 \to 2.19 \to 2.21\times$ and $1.46 \to 2.68 \to 2.78\times$) and shrinks on TensorCore ($2.06 \to 1.50 \to 1.57\times$; fair Tuner $1.71 \to 1.29 \to 1.36\times$), 90 layers per cell (stored canonical DeepSeek results, last re-run attempt).

Unconstrained search at a matched (larger) budget. The template-constrained baselines above search restricted spaces, as in the original Soter study, so their gap to CARTA conflates search quality with search-space restriction. To separate the two, we ran *unconstrained* random search and simulated annealing over the full Soter mapspace—same smart builder, same Timeloop model as CARTA—on a 90-layer batch-16 subset (30 layers per accelerator: 3 layers per workload at evenly spaced network depths, fixed before any control was run), giving each control 500 valid evaluator calls, 3× CARTA's median of 164 evaluator calls. Of the 90 layers, 69 belong to the 270-case suite and 21 are additional layers of DenseNet, ResNet50, ResNeXt50, and UNet for which no Tuner result exists, so every vs.-Tuner entry on this subset uses the 69 overlapping cases. A separate single-seed random@500 replicate over 90 Eyeriss, 90 Simba, and 18 TensorCore batch-16 layers (not in Figure 5) beats the latency-directed Tuner by $1.88\times$ case-weighted ($1.82\times$ with equal accelerator weighting; $1.76\times$ and $1.66\times$ against the fair Tuner), consistent with the full-suite 3-seed run's $1.82\times$ against the same latency-directed Tuner ($1.66\times$ against the fair Tuner, Table 9).

Two facts follow. First, on the three main accelerators the Tuner is a weak incumbent (Sec. 5): the same unconstrained random search at 500 evaluator calls *beats the fair Tuner* by $1.66\times$ geomean on the full batch-16 suite (and the latency-directed Tuner by $1.88\times$ on the 198-case replicate), so a large part of CARTA's $1.85\times$ margin over the Tuner is available to any reasonably strong search over the full mapspace. This does not hold on every accelerator: on NVDLALike random search, SA and the GA at 500 evaluator calls only tie the fair Tuner ("The Tuner, rerun fairly" below). Second, on this subset the canonical run is ahead of random search and SA at 500 evaluator calls (Table 8): $1.36\times$

Table 8: CARTA (median 164 evaluator calls) vs. unconstrained random search and simulated annealing (500 valid evaluator calls each) over the full mapspace on the 90-layer batch-16 subset. EDP, $\pm 1\%$ tie band (Sec. 4); geomean of baseline EDP / CARTA EDP (> 1 favors CARTA). CARTA is the canonical run; this subset table is superseded by the full-suite controls against the pooled runs (Figures 5 and 7).

Baseline (500 evaluator calls)	W	T	L	Non-loss	Geomean (Simba / Eyeriss / TC)
Random search	70	10	10	89%	1.36× (2.20 / 1.10 / 1.04)
Simulated annealing	59	13	18	80%	1.30× (1.95 / 1.06 / 1.06)

geomean over random search and $1.30\times$ over simulated annealing, concentrated on Simba ($2.20\times$ / $1.95\times$); on Eyeriss and TensorCore, unconstrained search at $3\times$ CARTA's median budget reaches a statistical tie (1.04–$1.10\times$). This subset reading is superseded by the full-suite controls against the pooled DeepSeek runs (Figures 5 and 7). There the structured GP and the strategy-seeded annealer reach parity at 500 evaluator calls ($0.995\times$ and $0.986\times$), and random search reaches parity at 2,000 ($1.01\times$; "Equal-wall-clock comparison" below).

The large margins over template-constrained baselines reflect their restricted search spaces. Against the canonical run on this subset, CARTA reaches equal-or-better quality than random search and SA with about a third of their evaluator calls (164 against 500). Against the pooled runs on the full suite, this advantage holds at CARTA's own budget ($1.21\times$ over random search, $1.06\times$ over the structured GP and the strategy-seeded annealer at 200 evaluator calls) and is gone for the structured controls at 500. CARTA is not a categorically stronger optimizer, and neither CARTA nor these searches train per task. Held to CARTA's own budget the gap widens (Table 11); the full-suite version of this control, with six optimizers, is Table 9 and Figure 7.

Full-suite budget-matched controls (270 cases, six optimizers). Table 9 and Figure 7 report the controls we consider primary. All run over the full Soter mapspace with the same builder and evaluator, on all 270 batch-16 cases, 3 seeds each, with best-so-far EDP recorded at 4–500 *valid* evaluator calls. They are: unconstrained random search; the paper's simulated-annealing baseline (identical warm-up, Metropolis rule, adaptive perturbation, and restart schedule); a GAMMA-style genetic algorithm (Kao & Krishna, 2020) (population 24, tournament selection, level-wise crossover with probability 0.7, mutation via the same perturbation operator with probability 0.5, elitism); a Tree-structured Parzen Estimator (TPE) surrogate optimizer (Optuna 5.0, multivariate, 10 start-up trials) over the same decision space, with spatial options and tile caps as categorical variables, loop priorities encoded as per-dimension scores whose ranking gives the permutation, invalid mappings returned as a penalty and only valid mappings counted; a Gaussian-process Bayesian optimizer (Optuna 5.0 `GPSampler`, log-EI acquisition, 20 start-up trials) over a *structured* encoding of the same decisions—one integer $\log_2$ exponent per dimension and spatial level, snapped to the nearest menu option so that neighbouring allocations are neighbours in the search space, priorities as per-dimension scores, tile caps as divisor indices—run to 500 valid evaluator calls; the same two surrogates re-run with the flat categorical encoding, to separate encoding from sampler family; and a *strategy-seeded annealer* that carries the same domain knowledge as CARTA without any LLM. For the annealer, the per-accelerator "recommended strategies" of the agent's system prompt (Sec. 3, reproduced in App. A) are transcribed mechanically into decision templates—each named spatial allocation snapped to the nearest option of the builder's menu, each recommended priority list and tile cap applied verbatim, every "or" alternative enumerated (8–40 templates per layer, median 13)—which are evaluated first, after which the paper's own simulated-annealing loop continues from the best of them.

The workload-level intervals of Figure 5 are percentile bootstraps over the 10 workloads, and with so few clusters the conclusion is method-dependent: an equal-weight cluster t-interval (per-workload mean log ratio, t_9) gives $[1.01, 1.13]$ for the strategy-seeded annealer, $[0.99, 1.14]$ for the structured GP, $[1.00, 1.17]$ for the structured TPE and $[1.12, 1.31]$ for random search, so the structured-search margins sit at the edge of significance under either method.

Like Table 9, the numbers in this paragraph and the next two are against the stored canonical run unless marked pooled. The 88-evaluator-call column illustrates a naive budget protocol: reading the controls at CARTA's prompt-database insertion count gives them about half of its median of 164 evaluator calls; the budget-matched reading at 150 and 200 evaluator calls, pooled over the

Table 9: Full-suite budget-matched controls (270 batch-16 cases, 3 seeds): geomean of best-so-far search EDP / CARTA EDP (> 1 favors CARTA) per accelerator and budget; W/T/L at $\pm 1\%$; "vs. Tuner" is fair Tuner EDP / search EDP (> 1 means the search beats the fair Tuner); the search/CARTA rows use the stored canonical run. The 200-evaluator-call column brackets CARTA's median of 164 evaluator calls; the 88-evaluator-call column is a half-budget reading obtained by treating database insertions as evaluator calls. In the 500 column, seed runs whose attempts yielded fewer than 500 valid mappings (73 random runs over 34 cases, 473–499 valid; 38 SA runs over 25 cases, 490–499) contribute their best-at-last-valid value, so all 270 cases are included and the curves are monotone. The Gaussian-process rows are taken entirely from the run extended to 500 valid evaluator calls (all 810 seed runs reached 500), which recorded no checkpoints at 300 and 400.

Valid evaluator calls	8	16	32	50	64	**88**	100	150	200	300	400	500
Random search												
Overall	2.76	2.12	1.76	1.62	1.55	**1.46**	1.42	1.34	1.29	1.24	1.21	1.18
Eyeriss	1.92	1.50	1.30	1.23	1.20	**1.16**	1.14	1.11	1.08	1.06	1.04	1.03
Simba	6.36	4.67	3.56	3.12	2.92	**2.62**	2.48	2.19	2.04	1.91	1.81	1.76
TensorCore	1.73	1.35	1.18	1.10	1.07	**1.03**	1.01	0.98	0.96	0.94	0.93	0.91
W/T/L	263/3/4	255/8/7	246/11/13	236/15/19	225/20/25	216/21/33	213/20/37	201/24/45	193/22/55	177/25/68	169/24/77	167/24/79
vs. Tuner	0.71	0.93	1.12	1.22	1.27	1.34	1.38	1.47	1.53	1.58	1.63	1.66
Simulated annealing												
Overall	5.25	3.45	2.52	2.10	1.90	**1.71**	1.62	1.45	1.34	1.23	1.16	1.12
Eyeriss	4.13	2.44	1.82	1.58	1.44	**1.32**	1.28	1.17	1.12	1.07	1.05	1.02
Simba	11.89	7.96	5.38	4.31	3.72	**3.11**	2.85	2.40	2.05	1.76	1.59	1.49
TensorCore	2.94	2.11	1.62	1.36	1.28	**1.20**	1.16	1.07	1.03	0.98	0.94	0.93
W/T/L	267/1/2	267/1/2	264/1/5	261/3/6	255/4/11	249/6/15	248/6/16	230/10/30	216/16/38	192/24/54	176/23/71	165/23/82
vs. Tuner	0.37	0.57	0.78	0.94	1.03	1.15	1.21	1.36	1.47	1.60	1.69	1.75
GA (GAMMA-style)												
Overall	2.75	2.12	1.77	1.62	1.54	**1.46**	1.43	1.34	1.30	1.23	1.20	1.17
Eyeriss	1.96	1.53	1.31	1.21	1.17	**1.12**	1.10	1.07	1.05	1.04	1.03	1.02
Simba	6.32	4.57	3.54	3.13	2.88	**2.62**	2.55	2.24	2.11	1.90	1.78	1.70
TensorCore	1.68	1.37	1.19	1.13	1.09	**1.06**	1.04	1.00	0.98	0.95	0.94	0.92
W/T/L	262/2/6	256/8/6	246/12/12	237/12/21	228/17/25	222/15/33	218/17/35	203/17/50	191/23/56	175/22/73	167/23/80	161/28/81
vs. Tuner	0.71	0.93	1.11	1.21	1.27	1.35	1.37	1.46	1.51	1.59	1.64	1.68
TPE surrogate optimizer												
Overall	2.73	2.10	1.72	1.55	1.47	**1.41**	1.38	1.33	1.30	1.29	1.28	1.27
Eyeriss	2.27	1.69	1.40	1.30	1.25	**1.19**	1.18	1.15	1.14	1.13	1.12	1.12
Simba	6.08	4.43	3.21	2.65	2.42	**2.23**	2.16	1.97	1.90	1.83	1.81	1.80
TensorCore	1.48	1.24	1.12	1.08	1.06	**1.04**	1.04	1.03	1.03	1.02	1.02	1.02
W/T/L	266/0/4	261/4/5	241/15/14	229/18/23	222/18/30	214/20/36	211/21/38	205/19/46	204/19/47	198/23/49	197/22/51	196/22/52
vs. Tuner	0.72	0.94	1.15	1.27	1.33	1.40	1.42	1.48	1.51	1.53	1.54	1.54
Bayesian optimization (GP surrogate; 500-call run)												
Overall	2.43	1.88	1.45	1.32	1.26	**1.22**	1.20	1.16	1.13	—	—	1.06
Eyeriss	2.19	1.65	1.30	1.22	1.19	**1.15**	1.13	1.10	1.08	—	—	1.01
Simba	4.72	3.43	2.28	1.91	1.75	**1.65**	1.60	1.52	1.47	—	—	1.34
TensorCore	1.40	1.17	1.03	0.99	0.97	**0.95**	0.94	0.92	0.90	—	—	0.87
W/T/L	261/3/6	252/4/14	228/14/28	210/20/40	200/18/52	186/27/57	183/26/61	171/28/71	163/30/77	—	—	130/39/101
vs. Tuner	0.81	1.05	1.35	1.49	1.56	1.62	1.64	1.70	1.74	—	—	1.86
Strategy-seeded annealer												
Overall	2.17	1.71	1.39	1.29	1.25	**1.20**	1.19	1.15	1.12	1.09	1.06	1.05
Eyeriss	1.79	1.48	1.35	1.25	1.20	**1.15**	1.14	1.10	1.07	1.05	1.02	1.01
Simba	3.14	2.12	1.66	1.53	1.49	**1.46**	1.45	1.40	1.38	1.33	1.28	1.25
TensorCore	1.81	1.58	1.20	1.12	1.08	**1.03**	1.02	0.98	0.96	0.94	0.92	0.91
W/T/L	252/10/8	240/13/17	223/19/28	216/21/33	212/22/36	203/23/44	201/20/49	183/24/63	175/28/67	165/27/78	153/30/87	140/32/98
vs. Tuner	0.91	1.15	1.42	1.52	1.58	1.64	1.65	1.71	1.75	1.80	1.84	1.87

three reproducible agent runs, is in Sec. 5. At 88 evaluator calls, random search and the GA are $1.46\times$ behind CARTA, SA $1.71\times$, and TPE $1.41\times$ (per-seed geomeans 1.43–1.49, 1.68–1.74, 1.43–1.50, and 1.37–1.43); the margin is 2.2–$3.1\times$ on Simba, 1.1–$1.3\times$ on Eyeriss, and 1.0–$1.2\times$ on TensorCore. With $3\times$ CARTA's budget (500 evaluator calls) the searches close to $1.18\times$ (random), $1.12\times$ (SA), and $1.17\times$ (GA) overall, reach parity on Eyeriss, and overtake CARTA on TensorCore (0.91–$0.93\times$), whereas TPE plateaus at $1.27\times$ (Eyeriss $1.12\times$, TensorCore $1.02\times$) after its early gains. Against the fair Tuner, random search passes it between 16 and 32 evaluator calls ($0.93\times$ at 16, $1.12\times$ at 32) and beats it by $1.66\times$ at 500 (SA $1.75\times$, GA $1.68\times$). Layer-level bootstrap 95% confidence intervals (2000 resamples over the 270 cases) are $[1.36, 1.57]$, $[1.59, 1.84]$, $[1.36, 1.56]$, and $[1.32, 1.51]$ for the four 88-evaluator-call margins, $[0.99, 1.07]$, $[1.14, 1.26]$, $[1.02, 1.09]$, and $[1.00, 1.09]$ for TensorCore@88 (random, SA, GA, TPE), $[0.94, 1.10]$, $[0.94, 1.09]$, $[0.93, 1.10]$, and $[1.02, 1.20]$ for Eyeriss@500, and $[1.12, 1.25]$, $[1.07, 1.19]$, $[1.11, 1.24]$, $[1.20, 1.35]$ at 500; per-accelerator W/T/L at 88 are Eyeriss 77/9/4, Simba 87/1/2, TensorCore 52/11/27 (random), 84/4/2, 90/0/0, 75/2/13 (SA), 74/6/10, 88/0/2, 60/9/21 (GA), and 77/6/7, 87/1/2, 50/13/27 (TPE).

The encoding, not the surrogate family, is what matters. Under the flat categorical encoding TPE and the Gaussian process are indistinguishable ($1.406\times$ and $1.403\times$ at 88 evaluator calls); under the structured encoding both improve to $1.185\times$ (TPE) and $1.215\times$ (GP), the best knowledge-free controls in the paper. The Gaussian process already *beats* CARTA on TensorCore at 88 ($0.95\times$, CI $[0.91, 0.98]$), and extended to 500 valid evaluator calls it closes to $1.057\times$ overall (CI $[1.006, 1.116]$;

Eyeriss $1.01\times$, Simba $1.34\times$, TensorCore $0.87\times$; W/T/L 130/39/101), and to parity against the pooled runs ($0.995\times$, Figure 7), at a median of 1,486 s per layer, the most expensive control we ran.

The strategy-seeded annealer matches it: $1.20\times$ behind CARTA at 88 evaluator calls (CI $[1.14, 1.27]$; Eyeriss $1.15\times$, Simba $1.46\times$, TensorCore $1.03\times$; W/T/L 203/23/44), $1.12\times$ at 200, and $1.05\times$ at 500 (CI $[1.00, 1.10]$; $0.986\times$ against the pooled runs, Figure 7), where it reaches parity on Eyeriss ($1.01\times$) and overtakes CARTA on TensorCore ($0.91\times$); against the fair Tuner it is $1.64\times$ at 88 and $1.87\times$ at 500; it passes the Tuner's final quality between 8 and 16 evaluator calls (random search between 16 and 32) and converges far faster thereafter. Comparing it with the knowledge-free controls separates the two ingredients of CARTA's budget-matched margin: the hand-written strategies account for the step from 1.41–$1.71\times$ down to $1.20\times$, and the remaining $1.20\times$ at 88 evaluator calls—$1.06\times$ at CARTA's own budget, pooled—is what the frozen LLM adds over search given the same prior at that budget; it reaches parity by 500. The same decomposition appears inside the agent, where removing the strategy text costs $1.19\times$ (Table 10). Per operator at 88 evaluator calls the residual is largest where the mapping has the most freedom (Simba depthwise $1.80\times$, GEMM $1.66\times$, grouped convolution $1.59\times$) and negative on the transposed convolutions that CARTA maps poorly (Eyeriss $0.36\times$, TensorCore $0.46\times$). The control costs a median of 263 s per layer for 500 evaluator calls, comparable to the other searches.

How the controls are implemented. All six controls use the builder's decision schema (App. G) but not quite CARTA's decision space. Each draws, per spatial level, a spatial allocation from the builder's enumerated menu of full-utilization factorizations, capped at 50 options; a loop-priority permutation per level and for DRAM; and optional tile caps drawn from the divisors of each dimension. CARTA's prompt instead lists up to 200 options, and CARTA may name others, including allocations outside the menu that 35 of the 270 traced CARTA winners use, some of them partial-utilization allocations realizable only under the traced run's builder revision ("Builder revisions" and the widened-menu rerun below). Every control attempt is evaluated (no deduplication), and only attempts the builder and Timeloop accept count as valid evaluator calls. SA is the paper's baseline implementation unchanged: up to 50 (then 100) random attempts to find a valid start, Metropolis acceptance on the EDP ratio with initial temperature 1.0 cooled by 0.995 per attempt to a floor of 0.01, perturbation strength growing with the non-improvement count, and a random restart with partial reheating after 100 non-improving attempts. The GA follows GAMMA: population 24, tournament selection, level-wise crossover with probability 0.7, mutation by the same perturbation operator with probability 0.5, and elitism of 2; at 88 valid evaluator calls it has completed only $\sim$3.7 generations, and SA at 88 attempts is still at temperature 0.64, which is why the GA tracks random search at CARTA's budget, while SA is *worse* than random below $\sim$300 evaluator calls and only overtakes it beyond that. Under our 20-worker harness, 500 valid evaluator calls take a median of 264 s (random), 245 s (SA), 179 s (GA), and 246 s (TPE) per layer, an upper bound on the uncontended cost, against CARTA's 266 s median (279 s mean) per layer in the traced run.

Where the budget-matched margin comes from. The margin over random search at 88 evaluator calls tracks the size of the decision space the builder exposes rather than the workload. Simba exposes two spatial levels (16- and 64-PE menus of up to 50 options each), five loop-priority lists, and per-level tile caps; Eyeriss two spatial levels of 16 PEs and five lists; TensorCore two spatial levels (4 and 256 PEs) but only three lists. The per-layer margin (random@88 / CARTA, geomean over the 3 seeds) is $2.62\times$ on Simba (82% of its 90 layers above $1.5\times$), $1.16\times$ on Eyeriss (14%), and $1.03\times$ on TensorCore (2%). Within Simba the ordering by operator is transposed convolution $3.9\times$ (3 layers), grouped convolution $3.6\times$ (9), standard convolution $2.8\times$ (39), GEMM $2.3\times$ (27), pointwise $2.1\times$ (8), depthwise $1.7\times$ (4): the more dimensions a layer has to place across the two spatial levels and five priority lists, the further random search is from the LLM's proposals at the same budget. On Eyeriss and TensorCore the three UNet transposed-convolution layers are the operator class where CARTA loses to random search on net (geomean $0.35\times$ and $0.64\times$; 2 of 3 and 3 of 3 layers lose, accounting for 2 of the 4 and 3 of the 27 per-case losses at 88 evaluator calls reported above; the largest non-T2D loss is TensorCore DenseNet-161 layer 80, $0.45\times$), the failure dissected in the case study of App. C (Table 19). By workload the margin ranges only from $1.29\times$ (DeepBench GEMMs) to $1.68\times$ (ResNeXt-50), so the accelerator, not the workload, is the dominant factor.

An error in the Eyeriss strategy text, as run. The agent's Eyeriss strategy text—part of the canonical agent, unchanged in every run reported here—describes the global-buffer and dummy-buffer fanouts as 168 and 14 and recommends spatial splits whose products are 168 and 14. The architecture's actual fanouts are 16 and 16 (256 PEs), and the auto-generated context prompt states them correctly, so the builder snaps the strategy text's suggestions to legal allocations. We report the agent as it was run. The strategy-seeded annealer's Eyeriss templates transcribe the same text, including splits with products 168 and 14, which the builder also snaps, so both sides of the Eyeriss comparison carry the error. The same kind of error in the first GemminiLike strategy text made the agent worse than no strategy text at all (Sec. 5), so a correct Eyeriss text could change the Eyeriss margins, and we have not run one. The small value of the text on Eyeriss (random EDP over the strategy-seeded annealer's is 1.01) may reflect this error rather than the accelerator.

A held-out accelerator. All three evaluated targets ship with the Soter framework and have a hand-written strategy text. To test whether the weak-incumbent finding is specific to them, we repeated the incumbent-versus-search comparison on a fourth accelerator outside the three evaluated targets, GemminiLike (held out of the main suite and of all strategy-text development, though the same 90 layers on it were used in May-2026 pilot campaigns with an earlier agent lineage and other prompts), a 16×16 systolic array with a three-level hierarchy (scratchpad, accumulator, PE register), on 30 layers (3 per workload) at batch 16. We ran the latency-directed Tuner with its standard 320-evaluator-call budget (163 s per layer) and the same four controls with 3 seeds each. This latency-directed incumbent is *weaker* here than on the three evaluated targets: random search is already ahead of it at the first checkpoint we record ($1.15\times$ at 4 valid evaluator calls, W/T/L 19/1/10; versus $\sim$16 on the main suite) and leads it by $1.44\times$ at 8, $2.10\times$ at 88, and $2.34\times$ at 500 evaluator calls; simulated annealing, the GA, and the Gaussian-process optimizer reach $1.86\times$, $2.08\times$, and $2.31\times$ at 88. These figures do not show that the RL Tuner's disadvantage against unconstrained search is a property of the tuner rather than of the accelerators. They use the latency-directed Tuner, and the held-out data against the fair Tuner contradict that reading. On GemminiLike random search passes the fair Tuner only at about 64 evaluator calls (about 100 for Tuner seeds 1 and 2) and leads it by 1.10–$1.14\times$ at 500 (end of this paragraph). On NVDLALike random search, SA and the GA at 500 evaluator calls only tie the fair Tuner, and only the structured GP beats it ("The Tuner, rerun fairly" below). How weak the Tuner is depends on the accelerator. We then ran CARTA itself there, where the agent receives only the auto-generated architecture description and none of the hand-written strategies that exist for the three evaluated targets. It beats the latency-directed Tuner ($1.76\times$) but loses to the fair Tuner ($0.86\times$) and to every control: at 88 valid evaluator calls the control EDP over CARTA's is 0.84 (random), 0.95 (SA), 0.85 (GA), and 0.76 (Gaussian process), and at 200—closer to the median 170 valid evaluator calls CARTA actually consumes here, i.e. about twice what the controls are given—0.78, 0.82, 0.78, and 0.74; per layer CARTA is better on 4 of 30, tied on 8, and worse on 18 against random search at 88. The run is smaller than the controls: an API budget was exhausted after 65 completed runs, leaving 25 layers with two seeds and 5 with three, and CARTA is noisy here (across the three-seed layers the largest and smallest EDP differ by a median of $1.45\times$); restricted to those five layers the picture is unchanged (0.89, 0.94, 0.77 for random, SA, and the Gaussian process at 88). All 25 missing runs are seed 2 (seeds 0 and 1 are complete): 10 failed part-way and 15 were skipped when the balance ran out, and none records an EDP, so none was excluded by outcome. Restricting every layer to its first two repetitions moves each ratio in this paragraph by at most 0.01: fair Tuner/CARTA $0.857\times$ (reported $0.861\times$), controls at 88 0.83/0.94/0.84/0.76, and at 200 0.78/0.81/0.78/0.73. This is one held-out accelerator and thirty layers: it falsifies our own portability claim, not the general capability of LLM proposers. Read with the strategy text ablation ($1.19\times$) and the strategy-seeded annealer ($1.20\times$), it places CARTA's budget-matched advantage in the hand-written per-accelerator prior rather than in the frozen LLM. *Objective-aligned comparison.* The Tuner figures so far in this paragraph are the latency-directed Tuner's, which optimized latency and folded batch into H for attention layers. Rerun with the EDP objective on CARTA's exact problems (same 320 evaluator calls), the Tuner is $2.04\times$ better on GemminiLike; against it the unported agent loses ($0.86\times$, CI $[0.77, 0.96]$, W/T/L 9/5/16) and the ported agent wins on the 20 test layers ($1.13\times$, CI $[1.03, 1.23]$, 12/7/1). Random search at 500 evaluator calls is $1.14\times$ better than the fair Tuner over the 30 layers (CI $[1.05, 1.25]$; $1.07\times$, CI $[0.99, 1.17]$, on the 20 test layers; $1.10\times$ and $1.11\times$ against Tuner seeds 1 and 2), so the incumbent is weak but not as weak as the latency-directed runs suggested. Against the fair Tuner, random search is behind at 4 evaluator calls ($0.56\times$) and at 32 ($0.91\times$), reaches it at 64 ($1.00\times$), and leads by $1.10\times$ at 200 (CI $[1.02, 1.21]$).

On the main suite it passes the fair Tuner between 16 and 32 evaluator calls. See "The Tuner, rerun fairly" below.

Porting to the held-out accelerator: protocol and pre-registration. The port of Sec. 5 was run with frozen artifacts, all included in the supplementary material. The development/test split and the strategy text were hashed before the test layers were run; the predictions were written, with a timestamp, to a plain-text file that is not hashed and to which the outcome was later appended, so their timing rests on that file and on the run logs rather than on a hash. (i) The 30 GemminiLike layers were split into 10 development and 20 test layers, stratified by workload, and the split was hashed before any diagnosis. The test layers were not untouched: the unported agent and the first strategy text (item iii) had already been run on all 30 layers before the split was frozen, so the split holds the test layers out of the diagnosis and of the second strategy text's development, not out of all prior exposure. The test set contains two pairs of identical layer shapes, so it has 18 distinct shapes. (ii) Diagnosis used the development layers only. GemminiLike is a 16×16 systolic array whose two spatial levels—the Scratchpad over 16 tiles and the Accumulator over 16 PEs—each carry a fanout of 16, while the PE register file has none. In 3 of the 10 development layers the best mapping found by random search assigns the full 16 to a single dimension at both levels (Scratchpad over N; Accumulator over C), and in 7 of 10 it does so at at least one of the two levels (5 of 10 if any of the stored top-5 mappings is counted). The agent instead splits each fanout over two or three dimensions, following the system prompt's generic advice that multi-dimensional spatial allocation beats single-dimensional—advice that suits Simba's 64-wide accumulation-buffer fanout. (iii) A first strategy text, written from the architecture file alone before this diagnosis, was *worse* than no strategy text ($0.85\times$, CI $[0.79, 0.91]$, 30 layers $\times$ 3 seeds, local backend): it placed the spatial allocation on the PE register file, which has no fanout. (iv) The second strategy text was written against the development layers and hashed. Three predictions were recorded before the test layers were run: P1, the ported agent beats random search at 88 evaluator calls; P2, it gains more than $1.15\times$ over the unported agent; P3, the per-layer correlation between search hardness and the agent's advantage turns positive. Outcome on the 20 test layers (60 runs, 3 seeds, DeepSeek-Chat, \$2.62): P1 holds ($1.15\times$, W/T/L 14/6/0). P2 holds ($1.34\times$, CI $[1.17, 1.58]$; see the qualifications below). P3 holds numerically (ρ from -0.33 to $+0.89$) but carries no information: it is a ratio-coupling artifact ("Search hardness" below), and we draw no conclusion from it. Comparisons against SA, the GA and the GP, and all 200-evaluator-call figures, were added after the outcome and are reported as such.

The ported result at its own budget. The ported agent consumed 152–200 valid evaluator calls on these layers (median 163, mean 172) against the controls' 88, so the 88-evaluator-call column gives it about $1.9\times$ the evaluator calls—the same asymmetry we correct for the *unported* result above, and we correct it here too rather than only where it is unfavourable. At the nearest measured checkpoint at or above the agent's own median (200 evaluator calls) the margins are 1.083 / 1.113 / 1.057 / $1.019\times$ against random / SA / GA / GP, and the "no losses" statement weakens by exactly one cell: the GP column becomes 6/13/1, i.e. a statistical tie with one loss, while random, SA and GA remain 10/10/0, 16/4/0 and 14/6/0. The full control sweep for the ported agent is $1.165/1.304/1.136/1.056$ at 64, $1.152/1.239/1.119/1.037$ at 88, $1.086/1.147/1.081/1.024$ at 150, $1.083/1.113/1.057/1.019$ at 200, and $1.049/1.044/1.024$ at 500. The unported agent at the same 200-evaluator-call checkpoint is $0.806/0.828/0.786/0.758\times$ ($0.802/0.823/0.782/0.754\times$ using only the first two repetitions of every layer, since that arm is incomplete; "Limits of the ported comparison"). The sign flip that the port is for therefore survives the matched budget—above parity against all four controls after the port, below parity against all four before it—at a cost of roughly $0.07\times$ of headline margin. These at-200 figures were computed as part of the pre-registration record and are reproduced here in full.

Limits of the ported comparison. Three qualifications. (i) The head-to-head against the unported agent rests on 44 of the 60 unported runs on these layers: the API exhaustion noted above leaves seed 2 present on only 4 of the 20 test layers. The per-seed figures $1.35/1.34/1.34$ are ported-seed-s against the *pooled* unported geomean, not three independently paired replications; the genuinely paired values are 1.325 (seed 0, 20 layers), 1.382 (seed 1, 20 layers) and 1.198 (seed 2, 4 layers only). We report the pooled form because seed 2 cannot carry a 20-layer pairing, and we state it here rather than letting the triple read as three replications. Restricting the unported arm to the first two repetitions of every test layer gives $1.35\times$ (CI $[1.18, 1.59]$) against the reported $1.34\times$ $[1.17, 1.58]$;

the 4 test layers with three repetitions give $1.27\times$ ($[1.08, 1.50]$), too few layers to be informative. (ii) The $1.34\times$ gain is concentrated: four layers (MobileNet L0 $3.07\times$, UNet L19 $2.58\times$, GPT-3 L5 $2.56\times$, GPT-1 L0 $1.78\times$; geomean $2.45\times$) carry most of it, the remaining sixteen geomean $1.157\times$, and 6 of the 20 sit inside the $\pm 1\%$ band. The direction is uniform—no layer is worse—but the magnitude is not. (iii) This is one port, one accelerator, one pre-registration; the falsified fanout prediction below is the evidence that we do not treat the recipe as general.

Port cost and exposure (from Sec. 5.4). GemminiLike also spent about 2,500 agent evaluator calls checking the frozen text on the development layers before the test run, after the first strategy text failed. The resulting text, about 30 lines, measures the artifact's size, not its acquisition cost. The split holds test layers out of diagnosis and strategy development, not all prior exposure: GemminiLike test layers had been run with the unported agent and the first strategy text before the split froze, and NVDLALike had appeared in unrelated runs; workload families are not held out, so these experiments test adaptation within the benchmark's workloads.

Where the port's strategy text comes from, and what that means for the comparison. The second strategy text is not an a-priori description of the architecture: it was written after looking at the best mappings random search found on the development layers (300 random evaluations on each of the 10 development layers, 3,000 in total), and its first strategy restates that structure (Scratchpad over $N{=}16$, Accumulator over $C{=}16$, per-level priorities, DRAM order and an Accumulator K cap); Scratchpad $N{=}16$ and Accumulator $C{=}16$ are each the top random-search mapping's allocation on 5 of the 10 development layers. The NVDLALike text was written the same way. Its 3,000 random evaluations on the 10 development layers were saved two minutes before the text was hashed, its Registers N caps (4–32) and spatial allocations restate those development winners, and the diagnosis compared the winners' per-level energy and cycles with the unported Qwen3-14B development mappings. Both port texts are therefore transcriptions of random-search winners on development layers plus the unported agent's development failures; neither was written a priori. The one text written from the architecture file alone, GemminiLike's first, made the agent worse. On 13 of the 20 test layers the ported agent returns the same EDP on all three seeds, consistent with it executing that template. We therefore ran the symmetric LLM-free control: the same text—about 30 lines of strategy text—transcribed into decision templates, exactly as for the main-suite strategy-seeded annealer, and continued with the same annealing loop on the same 20 layers, 3 seeds, 200 evaluator calls. The ported agent leads it by $1.045\times$ at 150 evaluator calls (CI $[1.013, 1.084]$) and $1.032\times$ at 200 (CI $[1.010, 1.059]$), with W/T/L 7/13/0 at both; the strategy-seeded annealer itself beats random search at 200 by $1.050\times$. Most of what the port buys is the text; the LLM adds a small, consistent increment.

Evaluator calls and database insertions. The traced batch-16 run counts a median of 164 evaluator calls per layer (mean 161.5, range 26–198) but only a median of 91 (mean 87.9) prompt-database insertions beyond the initial mapping. Controls count valid evaluator calls and are read at 200, an allowance, not exact per-case matching: 59 of the 540 September CARTA runs reach 200 counted calls, 201 with the initial mapping. The agent's evaluation counter increments once per Timeloop call, after de-duplication against the decision sets it has already tried; different decision sets can still build the same mapping. Over the 270 batch-16 cases of the traced full-suite run the counter's median is 164 (mean 161, range 26–198), and those calls build a median of 92 distinct mappings. Every control counts one evaluation per evaluator call, so 164 is the comparable figure. Counting prompt-database insertions gives $\sim$88 entries; the database suppresses insertions when the built mapping repeats a recent entry, so this counter underestimates evaluator calls. Every budget-matched comparison in this paper is therefore read at the controls' 150- and 200-evaluator-call checkpoints. Counting evaluator calls moves the headline spread over the eight controls from 1.19–$1.71\times$ to 1.12–$1.44\times$ for the canonical run and 1.06–$1.36\times$ for the traced run, and it applies in both directions: the held-out failure is 0.74–$0.82\times$ when the controls are given 200 rather than 88. The canonical run records no evaluation counts; we assume it spends like the traced run, which uses the same agent and call budget. Sample-efficiency figures elsewhere in this appendix that count 88 or "valid samples" count prompt-database insertions, not evaluator calls; measured in evaluator calls, CARTA uses 49% fewer than the Tuner's 320, not 72.5%.

The Tuner, rerun fairly. The latency-directed Tuner optimizes latency (it keeps the minimum-cycle candidate) while CARTA optimized EDP, and for batched-matmul (attention) layers the Tuner's driver folded the batch into the head dimension H while CARTA's folded it into N—the same MACs, different operand shapes, affecting 18 of the 270 batch-16 cases (the Tuner's folding is the faithful one; see "Attention-layer encoding" below). We reran the Tuner with two changes only: it optimizes EDP, and it solves exactly CARTA's problem specification. It keeps its 10 epochs $\times$ 32 candidates $=$ 320 evaluator calls; its internal evaluation pool was capped at four processes, which changes parallelism, not the algorithm. The Tuner is deterministic given its inputs and its fixed random seed (42): 330 cases that were run twice returned identical EDP. Because that seed fixes the policy initialization and sampling, we reran the fair Tuner with seeds 1 and 2 (the only change) on the 270 batch-16 cases and on both held-out accelerators. Against the pooled DeepSeek runs, fair Tuner/CARTA EDP is $1.85\times$, $1.80\times$ and $1.77\times$ for seeds 42, 1 and 2 (canonical run 1.97, 1.92, 1.88; pooled Qwen 1.61, 1.57, 1.54; random search at 500 evaluator calls 1.66, 1.62, 1.59), so the default seed is the one most favorable to CARTA, by up to 5%. Per case the Tuner varies by a median of $1.57\times$ across seeds, and the per-case best of the three seeds (960 evaluator calls) still has $1.43\times$ the pooled DeepSeek EDP ($1.24\times$ the pooled Qwen EDP). On GemminiLike the unported agent loses to every seed (0.83–$0.86\times$) and the ported agent beats every seed (1.11–$1.13\times$ on the test layers). On NVDLALike's 20 test layers the unported agents lose to every seed (DeepSeek 0.73–$0.79\times$, Qwen3-14B 0.60–$0.65\times$). The ported DeepSeek agent, the declared secondary arm, ties seed 42 ($1.02\times$, CI $[0.94, 1.10]$) and leads seeds 1 and 2 by $1.10\times$ (CIs $[1.012, 1.215]$ and $[1.000, 1.228]$; the seed-2 interval touches parity). The ported Qwen3-14B agent, the registered primary arm, loses to all three seeds: $0.670\times$ ($[0.565, 0.782]$), $0.723\times$ ($[0.605, 0.853]$) and $0.725\times$ ($[0.606, 0.859]$). All NVDLALike intervals here are layer bootstraps over the 20 test layers (2000 resamples). The final file of the ported Qwen3-14B arm has 59 of its 60 runs: it omits the seed-2 run on VGG-16 layer 11, which the arm's running result file recorded about one second after the final file was written; with that run included the seed-42 ratio is still $0.670\times$.

Re-selecting the minimum-EDP candidate from the latency-directed run's own 320 logged samples already improves it by $1.10\times$; the rerun improves it by $1.10\times$ at batch 16 and $1.12\times$ at batch 1. Against the fair Tuner, CARTA has lower EDP by $1.97\times$ at batch 16 for the canonical run (CI $[1.82, 2.13]$, W/T/L 240/10/20), $1.85\times$ pooled over the three later runs (CI $[1.72, 1.99]$, 246/6/18; Eyeriss 2.24, Simba 2.12, TensorCore 1.33), $1.58\times$ at batch 1 (CI $[1.47, 1.70]$; $1.55\times$ keeping first completed attempts), and $1.61\times$ with the pinned Qwen3-14B backend (CI $[1.49, 1.74]$). Over all 810 stored canonical results the lead is $1.85\times$ (CI $[1.77, 1.93]$, W/T/L 692/50/68; batch 64: $2.03\times$), against $2.05\times$ for the latency-directed Tuner; keeping each case's first completed attempt gives $1.82\times$ ("Re-runs in the canonical sweep"). Random search at 500 evaluator calls also beats the fair Tuner, by $1.66\times$ at batch 16 (all 270 cases; the 73 random seed runs that ended at 473–499 valid evaluator calls contribute their best at the last valid one).

On the held-out accelerators the win over the fair Tuner depends on the strategy text: on GemminiLike the unported agent loses to it ($0.86\times$) and the ported agent beats it ($1.13\times$ on the 20 test layers); on NVDLALike the fair Tuner is a competitive incumbent. Random search, SA and the GA at 500 evaluator calls only tie it: fair Tuner/control EDP is $0.944\times$ ($[0.81, 1.06]$), $1.005\times$ ($[0.91, 1.10]$) and $0.966\times$ ($[0.88, 1.06]$) on the 20 test layers, and $0.957\times$ ($[0.86, 1.04]$), $1.029\times$ ($[0.95, 1.11]$) and $0.995\times$ ($[0.90, 1.11]$) on all 30. Random search does not pass it at any checkpoint from 4 to 500 evaluator calls ($0.71\times$ at 88, $0.86\times$ at 200, all 30 layers). At 200 evaluator calls random search and SA lose to it ($0.84\times$ each on the test layers) and the GA nearly ties it ($0.92\times$, CI $[0.83, 1.01]$). The structured GP, which ran only to 200, is the one control that beats it there ($1.115\times$, $[1.05, 1.19]$, on all 30 layers; $1.050\times$, $[0.98, 1.12]$, on the test layers). Against Tuner seeds 1 and 2 the three controls at 500 tie or lead (1.01–$1.12\times$ on all 30 layers; only SA against seed 1 excludes parity). The unported agents lose to the seed-42 Tuner (DeepSeek $0.73\times$, $[0.63, 0.83]$; Qwen3-14B $0.60\times$, $[0.48, 0.72]$). The ported DeepSeek agent ties it ($1.02\times$, CI $[0.94, 1.10]$), and the ported Qwen3-14B agent, the registered primary arm, loses to it ($0.67\times$, CI $[0.57, 0.78]$).

Attention-layer encoding. The shared problem specification models batched attention imperfectly. CARTA's driver multiplies N by the batch size, while the second operand's projection (H, C, K) excludes N, so that operand is shared across the batch: BERT-Large layer 1 at batch 16 becomes $H{=}16$, $N{=}8192$, whereas independent batched attention needs $H{=}256$, $N{=}512$ (the Tuner driver's original folding). MACs are unchanged but reuse is not, so the six attention layers (18 cases per batch size) model a more reusable computation for every method alike. Excluding them

changes the headline ratios as follows: fair Tuner/CARTA over 810 cases $1.85 \rightarrow 1.83$ (batch 16, canonical, $1.97 \rightarrow 1.95$; batch 64 $2.03 \rightarrow 2.02$); pooled DeepSeek $1.85 \rightarrow 1.83$; pooled Qwen $1.61 \rightarrow 1.59$. Random, SA, structured GP and strategy-seeded annealer at 200 evaluator calls against pooled DeepSeek stay at 1.21, 1.26, 1.06 and 1.06; the widened-menu controls move from 1.19, 1.19, 1.05 to 1.20, 1.20, 1.05; structured GP and strategy-seeded annealer against Qwen move from 0.92, 0.92 to 0.93, 0.92. We did not rerun the attention cases with the faithful encoding.

Builder revisions and the Tuner's representation. Two revisions of Soter's smart builder were used. The traced May run used the earlier revision (repository commit `2714734`), and so, by date, did the earlier canonical sweep: rebuilding every traced winner from its logged decisions with that revision reproduces all 270 stored mappings exactly, whereas the current revision reproduces 239 and differs on 31, 23 of them in spatial factors. The earlier revision rounded a requested spatial factor to a divisor of the dimension only, so it realized partial fanout utilization; 23 of the 270 traced winners contain a non-power-of-two spatial factor (e.g., $C{=}3$ with $K{=}8$ and $N{=}2$ on Simba's 64-lane AccumulationBuffer for DenseNet-161 layer 0). The current revision also requires the factor to divide the remaining fanout, so on these power-of-two fanouts every spatial factor is a power of two; none of the 540 September winners contains another factor. All September runs (the two DeepSeek runs, the Qwen3-14B repetitions, the ports and the contamination control) and every search control used the current revision. Soter's Tuner does not use the smart builder: its program serializer builds spatial factors as powers of two, the same set the current revision allows. Restricting CARTA to the two September runs gives fair Tuner/CARTA $1.85\times$ (CI $[1.71, 1.99]$, W/T/L 243/11/16) and moves every control ratio at 200 evaluator calls by at most 0.01 (random 1.21, SA 1.26, GA 1.22, structured GP 1.06, structured TPE 1.08, strategy-seeded annealer 1.06; widened menus 1.19, 1.19, 1.05); the September pool's EDP is within 0.4% of the traced run's ($1.004\times$). The released snapshot contains both revisions and the rebuild script.

Decision space (from Sec. 5). The main controls enumerate at most 50 full-utilization allocations per spatial level, CARTA up to 200: of the 270 traced winners, 35 use allocations outside the controls' menu and 23 use partial fanout utilization, which only the traced run's earlier builder revision realizes (the current one cannot); restricting CARTA to the September runs moves every control ratio by at most 0.01. CARTA also suppresses duplicate decision sets, which these controls reevaluate. A rerun widens the menus of random search, annealing, and the strategy-seeded annealer to 200 options and deduplicates proposals (the structured GP and the port controls keep the original menus): at 200 evaluator calls, control/CARTA EDP is 1.19 [1.14, 1.25] for random search, 1.19 [1.14, 1.25] for annealing, and 1.05 [1.02, 1.08] for the strategy-seeded annealer, so the knowledge-free margin persists and the lead over the strategy-seeded annealer remains small.

Controls with widened menus. The menu-limited controls draw spatial allocations from the first 50 full-utilization factorizations per level and re-evaluate duplicate decision sets, whereas CARTA's prompt lists up to 200, the builder revision of the traced run accepted partial-utilization allocations (e.g. 48 of 64 lanes when $C{=}3$), and CARTA's counter skips duplicates. The current builder revision, used by this rerun and by every other control and the September agent runs, rounds each spatial factor to a divisor of the remaining fanout, so the partial-utilization options added below are rounded to full-utilization ones; the rerun therefore widens the full-utilization menu and aligns duplicate handling, not partial utilization.

We reran random search, simulated annealing and the strategy-seeded annealer with the first 200 full-utilization factorizations plus partial-utilization factorizations with product in $[T/2, T)$ (the two kinds sampled about equally often) and with CARTA's de-duplication, on all 270 batch-16 cases with 3 seeds. Against the pooled agent at 200 evaluator calls the margins move from 1.21 to $1.19\times$ (random, CI $[1.14, 1.25]$), from 1.26 to $1.19\times$ (SA, $[1.14, 1.25]$) and from 1.06 to $1.05\times$ (strategy-seeded annealer, $[1.02, 1.08]$). On Simba DenseNet-161 layer 0, where the stored canonical run's lead is largest, the same-space controls reach 1.50–1.52×10^{12} like the menu-limited controls, so that lead is not a menu artifact; the stored canonical run reached 1.31×10^{11} there, but the traced run reached 2.63×10^{11} and both September runs 1.50×10^{12}, the controls' value, so against the pooled agent (8.4×10^{11}) the lead on this layer is $1.8\times$ and the largest pooled-agent lead over same-space random search is Simba ResNeXt-50 layer 27 ($7.5\times$). The de-duplication history is per case: a history shared across cases would reject candidates seen only on another problem or seed and handicap the controls.

A pinned open-weight backend. To give the agent's side a confidence interval on a model that does not drift, we ran the unmodified agent on all 270 batch-16 cases with Qwen3-14B served locally (vLLM, thinking disabled), three repetitions. A second driver duplicated 85 of the seed-2 cases; we keep the primary driver's result for every case, and using the duplicates instead moves the fair-Tuner ratio from 1.61 to $1.60\times$. Per-seed margins agree within about $\pm 3\%$. Pooled, this agent has lower EDP than the fair Tuner by $1.61\times$ (CI $[1.49, 1.74]$), than random search by $1.06\times$ ($[1.01, 1.11]$) and than SA by $1.10\times$, but higher EDP than the structured GP ($0.92\times$, CI $[0.89, 0.96]$) and the strategy-seeded annealer ($0.92\times$, $[0.89, 0.95]$). Zero-training beats the per-task Tuner on both backends; beating structured search does not transfer from DeepSeek-Chat to the smaller model.

A second held-out accelerator: NVDLALike. NVDLALike has five memory levels and two spatial levels (16 and 4 wide, 64 PEs). The protocol (the same 30 layers and dev/test rule as GemminiLike, predictions recorded) and the split were fixed before the controls (2026-09-23: split hashed 13:11 UTC, controls 13:23–13:59, unported Qwen run 14:11); the strategy text was hashed at 23:18, after the controls and the unported run and before any ported-agent run (23:39–00:19). The NVDLALike model had been used in May-2026 agent and Tuner campaigns on the same 90 layers with an earlier agent lineage and other prompts, so the registration does not mark first exposure; the registered primary arm is the local Qwen3-14B backend, and a DeepSeek-Chat arm was added as a declared secondary analysis. Unported, the agent loses to every control on the 20 test layers (Qwen 0.57–$0.72\times$; DeepSeek 0.70–$0.87\times$); against random search, random/unported-agent EDP is 0.87 for DeepSeek and 0.72 for Qwen3-14B, the backend registered as primary before the port's agent runs (hollow circles in Figure 8). On the development layers the diagnosis was that on transformer GEMMs the agent leaves the batch-token dimension (N in the thousands) untiled at the innermost register level, where the best random-search mappings (300 evaluations per development layer) cap it at 4–32, which costs those layers $1.9\times$ in cycles. One strategy text was written from that diagnosis and those mappings and hashed before any test-layer run. With DeepSeek the ported agent beats random search, SA and the GA (1.22, 1.22, $1.11\times$) and ties the structured GP ($0.97\times$, CI $[0.94, 1.00]$), gains $1.40\times$ over the unported agent, and leads a strategy-seeded annealer by $1.05\times$ (CI $[1.00, 1.13]$). The registered Qwen arm gains $1.12\times$ from the port but still loses to every control (0.64–$0.80\times$), and the strategy-seeded annealer beats it by $1.45\times$: the registered prediction that the annealer would come within $1.05\times$ of the ported agent fails for the primary arm. Against the fair Tuner (seed 42) on the test layers, the ported DeepSeek agent ties ($1.02\times$, CI $[0.94, 1.10]$) and the ported Qwen3-14B agent loses ($0.67\times$, CI $[0.57, 0.78]$); per-seed values are under "The Tuner, rerun fairly".

Why we do not compare against Timeloop's native mapper. Timeloop's mapper refuses the unconstrained mapspaces of these accelerators (their size exceeds 2^{128}), and it still refuses them after the spatial dimensions are restricted to Timeloop's reference Simba dataflow; running it requires fixing temporal factors and loop permutations at several levels, and those choices would decide the comparison. We therefore compare against searches over the builder's decision space and, above, against the same searches with widened menus closer to CARTA's decision space.

Contamination control: anonymized architecture descriptions. Eyeriss, Simba and TensorCore are published designs, so we tested whether the margin is recall of published mappings. *Manipulation.* The design's name is replaced by a neutral token and every memory-level name—including the short forms and four-letter abbreviations the prompts use—by L0…Ln. The replacement is applied at a single point, to the complete text of every LLM call after all prompt builders have run, so the refine and polish phases (which quote the agent's own decisions and per-level energy breakdowns) are covered; the model's decisions are mapped back before the mapping is built, so the search space is unchanged. Capacities, fanouts and the tensors each level stores are left in place, and together they still identify Simba structurally: this control hides names, not structure. *Audit.* We ran the real agent through all three phases with a stub LLM that records exactly what it receives: with anonymization off it receives thousands of real-name tokens, with it on none, on all three accelerators. *Implementation defects.* The reported result is the third implementation; two earlier ones were withdrawn after their outcomes were known, and both would have met C1 spuriously. The first (2026-09-23) left the strategy text's short level forms (e.g. "AccBuf") unreplaced. An interim look at it gave fully-anonymized/named $0.796\times$ (CI $[0.671, 0.927]$, 21 layers; $0.827\times$, $[0.70, 0.95]$, on the 24 layers it completed, against the second implementation's named arm), and the name-only arm

was added after that look. The second (same day) anonymized individual prompt builders and left names in the refine and polish phases. It gave fully anonymized $0.865\times$ ($[0.747, 0.993]$) and name-only $1.014\times$ ($[0.869, 1.177]$), and the record's verdict at the time ("C1 holds on the point estimate") was withdrawn on 2026-09-24, after an external reviewer found the leaks. Both measured a partially anonymized interface and neither is used. All three arms were then re-run on 2026-09-24 with the single-point replacement described above, and the outgoing-text audit above checks the complete interface. The registered predictions were C1 (recall if fully-anonymized/named < 0.90), C2 (structure if within $[0.95, 1.05]$) and C3 (the anonymized arm still beats random search and annealing). *Result* (DeepSeek-Chat, same day, 30 Simba batch-16 layers, one run per arm; one name-only run failed, so that comparison has 29 pairs). Ratios are named-arm EDP / manipulated-arm EDP, so above 1 means the manipulated arm found lower EDP. Name-only against named is $1.110\times$ (CI $[0.974, 1.286]$); fully anonymized against named is $1.104\times$ (CI $[0.959, 1.280]$). C1 is not met: hiding every name does not make the agent worse. C2 is not met on the point estimate either (1.104 lies outside $[0.95, 1.05]$), though its interval overlaps that range. Fully anonymized, it leads random search by $1.64\times$, SA by $1.49\times$, the structured GP by $1.33\times$ (CI $[1.11, 1.62]$) and the strategy-seeded annealer by $1.26\times$ (CI $[1.10, 1.45]$). The anonymized arms spent more evaluator calls (median 195 and 183 against 168), which favours them; the control covers one accelerator at one seed.

Search hardness as an amplifier. *Not evidence.* Define per-layer search hardness as random search's best at 88 evaluator calls over its best at 500. It correlates with the agent's advantage over random search at 88 ($\rho = +0.565$ over 284 layers; on the held-out test layers -0.33 before the port and $+0.89$ after, the flip registered as P3). Both quantities share random search's 88-evaluator-call result, so any method that reaches roughly random@500 quality produces a high positive correlation: on the same 20 test layers, substituting random search at 200, the GP at 200, the GA at 200 or SA at 500 for the agent gives $\rho = +0.94$, $+0.93$, $+0.81$ and $+0.84$. The correlation says nothing about the agent's prior beyond P1, and we draw no conclusion from it.

A stronger prediction, falsified. If the unported failure were caused by a narrow fanout, the deficit should shrink as the fanout widens. We pre-registered that prediction and ran the same accelerator family at fanouts 8, 16 and 32 (GemminiLikePE64, GemminiLike, GemminiLikePE1024; 20 test layers $\times$ 3 seeds each, one backend, no-addendum arm). It is falsified in the opposite direction: the agent scores 0.889 (CI $[0.83, 0.94]$), 0.806 ($[0.72, 0.89]$) and 0.691 ($[0.59, 0.80]$) against random search at 88 evaluator calls. Two trivial explanations are excluded: search hardness is flat across the family (1.111, 1.113, 1.104) and the agent obtains the same number of valid evaluator calls everywhere (median 157, 157, 158), so the degradation is in proposal quality. Post hoc, the agent's shortfall tracks how rarely it gives a spatial level its full width on one dimension—32%, 12%, 7% of spatial levels against the winners' 78%, 78%, 30%—which suggests that a wider fanout simply offers more ways to fragment. We label this reading post hoc; the prediction it generates (a single-dimensional correction should recover *more* at wider fanout, the opposite of the registered prediction) is untested.

The margin measured as budget. Figure 5 reports ratios at a fixed budget; the same runs can be read as the budget a control needs to reach CARTA's result. Extending random search to 2000 evaluator calls ($12.2\times$ CARTA's median of 164 evaluator calls) on all 270 batch-16 cases, we take per case the median over the 3 seeds of whether random's best-so-far at 2000 comes within $\pm 1\%$ of CARTA's EDP (the paper's tie band). Against the canonical run, random never reaches CARTA on 81% of Simba layers, 46% of Eyeriss layers and 10% of TensorCore layers; against the traced run on 61%, 48% and 7%. Under a strict inequality instead of the tie band the canonical-run rates are 86% / 54% / 18%; 15 of the 270 Simba seed-runs finish inside the band above CARTA, i.e. tied rather than beaten. For the canonical run, simulated annealing at 500 fails to match on 88% / 67% / 32% and the structured GP at 200 on 83% / 77% / 21% (strict inequality). CARTA's EDP here is a single run stored to three significant figures ($\pm 0.26\%$ quantization, within the tie band), which is why we report both full-suite runs.

Run-to-run variation of the agent. The canonical CARTA runs are single runs, so we bound their sampling noise with three-seed repetitions on a local open-weight backend (Qwen3-14B, 30 layers $\times$ 3 seeds per arm). Against random search at 88 evaluator calls the per-seed margins are $1.438/1.353/1.403$ on Simba with the strategy text and $1.407/1.423/1.400$ without it—a spread of

6.1% and 1.6% of the margin—and $0.811/0.750/0.731$ on GemminiLike before the port, a spread of 10.4%, the largest of the three. Across all eight multi-seed agent arms in our artifact the implied margin spread runs from 0.7% (20 GemminiLike test layers, ported) to 10.4% (30 GemminiLike layers, unported), so the correct envelope is 0.7–10.4% rather than the 1.6–6.1% that the two Simba arms alone would give; per target it is 1.6–6.1% on Simba and 10.4% on the held-out one. Two limits of this bound should be read with it. First, it is an *aggregate* bound: per layer the across-seed dispersion is much larger, with a median max/min EDP over three seeds of 1.00 on Simba (geomean 1.11–1.28, worst layer $4.4\times$) but 1.50 on GemminiLike, where 15 of 30 layers move by more than $1.5\times$. Sampling noise is small for the 30-layer geomean, not for an individual layer. Second, these particular repetitions were run on the Qwen3-14B backend on Simba and the GemminiLike family; the full-suite repetitions reported in the main text (three Qwen3-14B repetitions and three DeepSeek runs over all 270 batch-16 cases) supersede them as the estimate of run-to-run variation for the headline. Even at 10.4%, sampling noise stays well below the drift across model versions ($1.86\times$ vs. $3.03\times$ on 17 shared layers, a 63% move; this appendix, "Which run is which"), which is the reproducibility risk we flag; it is about $8\times$ below the 85% headline margin over the fair Tuner but only half the 21% margin over random search at 200 evaluator calls, so we do not claim that sampling noise is negligible against the search margins.

What the strategy text is worth, measured against the controls. The mechanism ablation reports that deleting the strategy text costs $1.19\times$ *relative to the agent with it*. Measured against the matched-budget controls instead, the ablated agent is still ahead of the three sampling searches, but the correct budget matters and we state it explicitly. The no-addendum arm ran on 34 of the 90 Simba batch-16 layers—the stratified prefix of Table 10, truncated by API-budget exhaustion and therefore skewed to early layers (layer ids $\{0, 1, 2, 3, 4, 5, 10, 32, 40, 64\}$ over 10 workloads)—and consumed a median of 170 valid evaluator calls (mean 176, range 155–200), so the control checkpoint that matches its own spend is 200, not 88. At 200 it leads random search by $1.15\times$ (CI $[0.96, 1.31]$; W/T/L 21/6/7), SA by $1.14\times$ ($[0.93, 1.32]$; 20/6/8) and the GA by $1.23\times$ ($[1.00, 1.46]$; 19/6/9): above parity against all three, with intervals that touch it. At the controls' 88 evaluator calls—roughly half its own spend—the same margins read $1.45\times$, $1.71\times$ and $1.53\times$, and at 150 (its median spend rounded down to a measured checkpoint) $1.22/1.31/1.33\times$. The stronger controls close the gap or pass it, and we report them rather than only the three weakest. On the same 34 cases the ablated agent is $1.19\times$ ahead of flat-encoding TPE and $1.36\times$ ahead of flat-encoding GP at 88, but already behind the structured-encoding surrogates and the strategy-seeded annealer there (TPE-structured $0.89\times$, GP-structured $0.95\times$, strategy-seeded annealer $0.90\times$); at the matched 200 it is behind all three ($0.83\times$, $0.88\times$, $0.88\times$) and a dead tie with flat TPE ($1.00\times$). This is the expected shape: strip the hand-written prior and what remains is a competent but not dominant proposal distribution, which is exactly the decomposition Sec. 5 argues for. It is not a retreat from the paper's claim, which is that a frozen model given a strategy text beats a per-(layer, accelerator)-trained tuner on the main suite, not that it beats every optimizer; on NVDLALike the ported Qwen3-14B agent loses to that tuner and the ported DeepSeek agent ties it. On the Qwen3-14B backend the strategy text is worth $0.99\times$ (CI $[0.86, 1.11]$), i.e. nothing. That arm consumed a median of 155 evaluator calls (range 146–169; the median of 164 in Sec. 4 is measured on DeepSeek), so we give its margin at both ends: against random search it leads by $1.41\times$ at 88, $1.31\times$ at 150, $1.23\times$ at 200 and $1.08\times$ at 500, and against SA by $1.52\times$ at 88 and $1.12\times$ at 200, but SA reaches parity at 300 ($1.00\times$) and overtakes it at 500 ($0.89\times$, W/T/L 6/11/13). Against random search the direction is unchanged at every budget we measured; against SA it reverses beyond 300 evaluator calls. The backend identity of this arm is recorded in our run metadata and directory structure rather than in the per-case CSVs, which do not carry a model id. On the accelerators it was written for, the prior's measured value therefore depends on the backend and the accelerator: $1.19\times$ for DeepSeek on 17 paired Simba layers, $0.99\times$ for Qwen3-14B on 30 Simba layers, and, measured through the strategy-seeded annealer, $1.49\times$ over random search on Simba but 1.01 and 1.00 on Eyeriss and TensorCore. On the ports it turns DeepSeek's losses to knowledge-free search into wins (Sec. 5.4).

Mechanism ablation in brief. Six months after the canonical run we re-ran the unmodified agent and two ablated variants on Simba batch-16 layers with DeepSeek-Chat as served in September 2026 (Table 10; protocol under "Independent rerun and mechanism ablation" below, App. C). Paired against a same-day control, removing the strategy text costs $1.19\times$ (paired t-test on log ratios, p=0.02; Wilcoxon p=0.04) while scalar EDP feedback shows no detectable harm ($0.94\times$, the ablated arm better on 6 of 16 layers and worse on 2; paired t-test p=0.42); both variants still beat the

Tuner, and the same-day control beats same-day random, SA, GA and TPE by 1.53–$1.98\times$ at 88 evaluator calls, so the margin is not an artifact of the canonical run.

Equal-wall-clock comparison. An equal-wall-clock protocol gives searches CARTA's elapsed time rather than its evaluation count. CARTA's measured per-layer time is 279 s in total, including its own analytical-evaluator calls. We treat all of it as non-evaluator overhead, which slightly favors the searches: for an evaluator that costs c seconds per call, CARTA takes $279 + 164c$ seconds and a search given the same time runs $N(c) = 279/c + 164$ evaluator calls. Reading the best-so-far curves at $N(c)$ (log-interpolated between checkpoints, capped at the measured 500), the four knowledge-free searches (random, SA, GA, flat TPE) remain at least 1.12–$1.27\times$ behind the canonical run overall for every $c \geq 0.83$ s (the reading at $c = 0.83$ s, where $N = 500$; Simba 1.49–$1.81\times$, TensorCore 0.91–$1.02\times$), and the margin grows with c: at $c = 3$ s ($N \approx 257$) they are 1.26–$1.29\times$ behind and at $c = 10$ s ($N \approx 192$) 1.30–$1.35\times$; the strategy-seeded annealer is 1.05–$1.11\times$ behind over $c \in [0.83, 3]$ s and $1.13\times$ at $c = 10$ s. TensorCore does not clearly favor CARTA at any evaluator cost. Against the traced run all of these ratios are about $1.06\times$ lower. The structured GP is omitted here because its checkpointed curves in this protocol stop at 200 evaluator calls (its separate 500-evaluator-call run is reported above).

For cheaper evaluators the searches receive more than 500 evaluator calls, so we extended random search to 2,000 valid evaluator calls per layer ($12.2\times$ CARTA's budget; 3 seeds, all 270 cases). Against the canonical run it is still $1.07\times$ behind overall (CI $[1.02, 1.13]$; W/T/L 133/37/100) and $1.12\times$ at 1,000; against the pooled runs it reaches parity ($1.01\times$, CI $[0.97, 1.05]$, W/T/L 108/46/116; $1.05\times$ at 1,000). Against the canonical run the accelerators separate: random search passes CARTA on TensorCore at $\sim$150 evaluator calls ($0.88\times$ at 2,000) and on Eyeriss at $\sim$1,000 ($0.98\times$ at 2,000), while Simba alone keeps the overall geomean above parity ($1.42\times$ at 2,000). Timeloop costs ~ 0.5 s per mapping under our 20-worker harness, so 2,000 evaluator calls take $\sim$1,010 s per layer against CARTA's 279 s; at an uncontended $\sim$0.1 s CARTA's wall-clock would buy $\sim$2,900 random evaluator calls, slightly beyond our measured curve, where the overall margin would be a few percent above parity and CARTA would retain its advantage only on the deepest hierarchy.

Four full-suite runs of the agent. All four use the unmodified agent (its system and context prompts are byte-identical across the runs), DeepSeek-Chat as served on the run date, 40 calls per layer and the 270 batch-16 cases: the canonical run (March 2026), the traced run (May 2026), and two runs on 2026-09-23 (seeds 1 and 2; the seed labels only order the runs, since the API is not seeded). No per-run file of the two September runs records the served model: their case logs and result files carry no model, provider or endpoint string, and their identity rests on the driver source, which requests the provider's default `deepseek-chat` alias, i.e. whatever the API served on the run date. The two September runs median 170 evaluator calls and 95 prompt-database insertions per layer. Against the controls at 200 evaluator calls the three later runs give 1.21–1.26 / 1.22–1.27 / 1.20–$1.24\times$ over the knowledge-free searches and 1.06–1.08 / 1.07–1.09 / 1.05–$1.06\times$ over the structured controls; the canonical run gives 1.29–$1.34\times$ and 1.12–$1.14\times$. The two same-day runs differ by $1.02\times$ in geomean EDP, and 166 of 270 layers agree within 5%. Pooled statistics use the per-layer geomean of the three later runs and a bootstrap over layers.

Which run is which. Four full-suite DeepSeek-Chat runs of the 270-case batch-16 suite exist (the two September runs are described under "Four full-suite runs of the agent"); the first two are the canonical run (March 2026; fair Tuner/CARTA EDP $1.97\times$, 240/270 wins, Simba $2.68\times$; Table 1), designated before any other run was made, and a traced run (May 2026; $1.85\times$, 238/270, Simba $2.20\times$) that supplies the per-call token, evaluator-call, and time accounting of Tables 14–15. Per layer, the traced run's EDP over the canonical run's has median 1.00 (135 of 270 layers within $\pm 5\%$), with 13 layers more than $2\times$ worse and 4 more than $2\times$ better, i.e. the run-to-run drift is heavy-tailed but symmetric enough that the two runs' geomeans differ by only $1.06\times$. The September-2026 mechanism-ablation arm (Table 10; 17 Simba layers) and the GPT-4o-mini and Qwen3.5 runs (May 2026) are the remaining runs; no other full-suite run was made or discarded. The batch-1 and batch-64 columns of Table 1 come from the same canonical March-2026 sweep, in which 281 of the 810 cases were re-run (next paragraph); only the batch-16 suite carries the controls, so every matched-budget number in this paper is batch-16.

Re-runs in the canonical sweep. The canonical sweep is not one attempt per case. Its repair scripts re-ran every case whose CARTA EDP exceeded the latency-directed Tuner's by more than $1000\times$ and then, in three further passes, by more than $3\times$, overwriting the stored result. We reconstructed every attempt from the sweep and repair logs. Of the 810 cases, 281 were re-run. All 261 cases flagged at $1000\times$ have a logged first attempt that completed zero evaluator calls: every API call failed and the result was the builder's initial mapping. The $3\times$ passes touched 24 cases: the preceding attempt had completed zero evaluator calls in 15 and 5–49 evaluator calls in 6 (API failures part-way through the run), and is not logged in 3; in one of those three (Simba, MobileNet layer 11, batch 1) a completed 146-evaluator-call re-run was itself replaced by a $9.5\times$ better one. Because the $3\times$ trigger conditions on the outcome, we report the canonical sweep keeping each case's first attempt that completed at least one evaluation: fair Tuner/CARTA EDP is $1.82\times$ over the 810 cases (CI $[1.74, 1.91]$, W/T/L 691/49/70; batch 1 $1.55\times$, batch 16 $1.97\times$, batch 64 $1.98\times$), against $1.85\times$ for the stored results; counting an attempt only if it completed at least 100 evaluator calls gives $1.84\times$. The pooled batch-16 runs contain no re-runs: every September run completed (at least 146 evaluator calls) and every traced case issued all 40 LLM calls. The single-run budget rows of App. J were produced with the same repair scripts.

Budget-matched random search on the 90-layer subset (March-2026 control). To separate CARTA's sample efficiency from its search budget, we ran unconstrained random search over the full mapspace on the same 90-layer batch-16 subset with 3 independent seeds (200 trials each; the 500-evaluator-call column is the single-seed control above) and recorded the best-so-far EDP after each of 4, 8, 16, 32, 50, 64, 88, 100, 150, 200 valid evaluator calls. Table 11 reports the geomean over layers of (seed-geomean random best-so-far EDP)/(CARTA final EDP). At 88 evaluator calls, about half CARTA's median spend, random search is $1.72\times$ behind CARTA overall (Simba $3.38\times$, Eyeriss $1.30\times$, TensorCore $1.16\times$; W/T/L 80/9/1); the per-seed overall geomeans at 88 evaluator calls are 1.70–$1.74\times$, so the estimate is not seed-driven. The curve decays slowly: $1.53\times$ at 200 and $1.36\times$ at 500 evaluator calls, and it never reaches $1.0\times$ within the $5.7\times$-larger budget. Random search reaches the Tuner's final quality after only 16 valid evaluator calls (last row), which quantifies how weak the latency-directed incumbent is; on this subset the canonical run is ahead of random search at every budget on every accelerator, and the gap is largest on the deepest hierarchy (Simba). This subset control is superseded by the full-suite controls (Table 9), where random search passes the canonical run on TensorCore at about 150 evaluator calls and reaches parity with the pooled runs at 2,000 ("Equal-wall-clock comparison"). The Simba margin is heavy-tailed: on three first-layer cases (DenseNet-161, ResNet-50, ResNeXt-50, layer 0) the canonical run found mappings $11\times$ better than random or SA at 500 evaluator calls; these mappings re-score identically under `timeloop-model` (the DenseNet-161 one is dissected in Table 19 of App. C), and two later independent runs found a mapping of the same quality on the DenseNet layer, but the September-2026 reruns of Table 10 did not. Excluding these three cases the subset's budget-matched margin is $1.61\times$ overall (Simba $2.96\times$, Eyeriss $1.30\times$, TensorCore $1.16\times$) and its 500-evaluator-call margin is $1.26\times$; the corresponding full-suite figures are given above.

Independent rerun and mechanism ablation. Table 10 reports three arms run on 2026-09-17 with the DeepSeek-Chat model served on that date, the unmodified agent code, the same 40-call budget and the same builder and evaluator, on a stratified prefix of the Simba batch-16 suite (the first layers of every workload in round-robin order; the run stopped when the API budget was exhausted). A run is included if its log records no failed API call; once the budget was exhausted every remaining run returned the builder's initial mapping after about 18 s, and those runs are excluded. This leaves 17 control, 34 no-strategy-text and 34 no-breakdown runs. The `status` column of the released `results.csv` is inverted by a driver bug (it marks the exhausted stubs `OK`); the README states the rule applied here. *Control* is the unmodified agent; *no addendum* removes the per-accelerator "recommended strategies" strategy text from the system prompt; *no breakdown* replaces the per-level energy breakdown in the refine/polish feedback by the scalar EDP alone. Because the served model differs from the canonical run's, the ablation arms are compared with the same-day control on the layers they share (paired geomean of arm EDP / control EDP; W/T/L at $\pm 1\%$), not with the canonical run.

The control itself beats the Tuner by $1.86\times$ on its 17 layers, where the canonical run scores $3.03\times$; per layer, the control is within $\pm 5\%$ of the canonical run on 9 of 17 layers (median ratio 1.02) and more than $2\times$ worse on 4 (the DenseNet-161 and ResNet-50 first-layer cases above—ResNeXt-50

Table 10: Mechanism ablation on Simba batch 16 (2026-09-17, DeepSeek-Chat as served on that date; 40 LLM calls per layer). Paired columns compare each arm with the same-day control on shared layers (geomean of arm EDP / control EDP, > 1 means the ablation hurts; W/T/L counts layers where the arm is better/tied/worse at $\pm 1\%$). "vs. Tuner" is the geomean *latency-directed* Tuner EDP / arm EDP on the arm's layers, with the canonical run's value on the same layers in parentheses; it is kept for reference only, and the paired columns carry the evidence. "Layers" is the number of layers each arm was run on; the paired columns use the 17 (no addendum) and 16 (no energy breakdown) layers shared with the control.

Arm	Layers	Paired vs. control	W/T/L	vs. Tuner (canonical)
Control (unmodified agent, Sept. 2026)	17	1.00×	—	1.86× (3.03×)
No addendum	34	1.19× worse	1/7/9	1.65× (2.99×)
No energy breakdown (scalar EDP feedback)	34	0.94× (not worse)	6/8/2	2.03× (2.81×)

Table 11: Budget-matched control: best-so-far unconstrained random search vs. CARTA's final result on the 90-layer batch-16 subset (3 seeds; geomean of random EDP / CARTA EDP, > 1 favors CARTA; W/T/L at $\pm 1\%$). CARTA spends a median of 164 evaluator calls, so the 150- and 200-evaluator-call columns are the budget-matched ones; the 88-evaluator-call column is a half-budget reading obtained by treating database insertions as evaluator calls. †500-evaluator-call single-seed control of Table 8. Last row: latency-directed Tuner EDP / random EDP (> 1 means random search beats the Tuner). CARTA is the canonical run; superseded by the full-suite controls of Table 9 and the pooled-run comparison of Figure 7.

Valid evaluator calls	8	16	32	50	64	**88**	100	150	200	500†
Overall	3.15	2.34	1.96	1.85	1.81	**1.72**	1.68	1.59	1.53	1.36
Eyeriss	2.18	1.67	1.42	1.37	1.34	**1.30**	1.28	1.25	1.21	1.10
Simba	7.09	4.99	4.15	3.84	3.71	**3.38**	3.27	2.95	2.74	2.20
TensorCore	2.02	1.53	1.28	1.21	1.20	**1.16**	1.12	1.10	1.08	1.04
W/T/L	88/2/0	87/3/0	86/3/1	84/5/1	84/5/1	**80/9/1**	80/8/2	79/6/5	75/7/7	70/10/10
Random vs. latency-directed Tuner	0.85	1.14	1.32	1.39	1.42	1.49	1.52	1.61	1.68	1.85

layer 0 was not among the 17—and two BERT-Large GEMMs, layers 0 and 3). The September arms record a median of 170 evaluator calls per 40 LLM calls, against 164 in the traced May run (the canonical run records none), so the ablated arms did not search less.

On the same 17 layers, same-day random search, SA, GA, and TPE at 88 valid evaluator calls (3 seeds) are $1.87\times$, $1.98\times$, $1.92\times$, and $1.53\times$ behind the September control (W/T/L 15/2/0, 15/2/0, 15/2/0, 14/2/1); because the control consumed a median of 170 valid evaluator calls, we also report the searches at 150 and 200 evaluator calls: $1.45/1.56/1.74/1.35\times$ and $1.37/1.37/1.55/1.29\times$ (12–15 wins of 17 at 150; 11–15 at 200). At 200 evaluator calls the same-day control thus leads the knowledge-free searches but not, distinguishably, the three controls that carry structure: structured GP $1.10\times$ (CI $[0.95, 1.27]$), structured TPE $1.01\times$ ($[0.85, 1.19]$), strategy-seeded annealer $1.06\times$ ($[0.93, 1.24]$). On these 17 layers the margin over the structured GP falls from the canonical run to the traced run to this rerun, which is the model-version drift described under "Which run is which".

C.3 Per-Operator Batch Scalability

The batch-size effect of Sec. C.2 varies substantially across operator types. Figure 11 presents the per-operator-type EDP ratio of the latency-directed Tuner to the stored canonical DeepSeek results (last re-run attempt) across batch sizes 1, 16, and 64 on Simba, for the five operator types with more than one layer (GEMM, C2D, T2D, DEP, PW); the single GRP layer is reported in the text.

Per-operator ratios by batch. The Simba geomean ratio (latency-directed Tuner EDP / stored CARTA EDP) at batch 1/16/64 is C2D $1.65/3.58/4.19$ (45 layers), GEMM $1.55/2.65/2.38$ (29), PW $2.01/2.51/2.52$ (8), GRP $1.04/1.22/1.65$ (1 layer, not plotted), DEP $0.61/1.03/0.95$ (4), and T2D $0.89/2.27/1.17$ (3); over all 90 layers it is $1.54/2.90/2.96$. C2D, PW, and GRP rise monotonically with batch, and C2D has the largest gain from batch 1 to 64; GEMM, DEP, and T2D peak at batch 16 and fall at batch 64, T2D from $2.27\times$ to $1.17\times$. CARTA is a net loss on DEP and T2D at batch 1 and on DEP at batch 64.

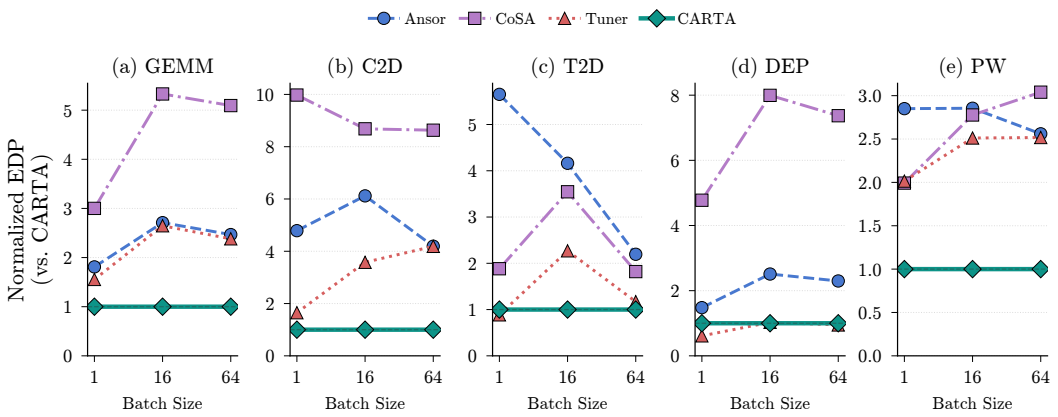

Figure 11: Per-operator batch scalability on Simba, stored canonical DeepSeek results (last re-run attempt) vs the latency-directed Tuner. Each panel plots, for one operator type, the EDP of the latency-directed Tuner (and of Ansor and CoSA) normalized to CARTA's at batch 1, 16, and 64; values > 1 indicate CARTA wins. Against the Tuner the ratio rises with batch on C2D and PW and peaks at batch 16 on GEMM, DEP, and T2D; DEP at batch 1 and 64 and T2D at batch 1 are net losses.

Table 12: Win/tie/loss of CARTA against the fair Tuner per configuration (90 layers each, $\pm 1\%$ band; canonical sweep, first completed attempt per case), with the geomean fair Tuner/CARTA EDP ratio in parentheses.

Accelerator	Batch 1	Batch 16	Batch 64
Eyeriss	81/5/4 (1.55)	89/0/1 (2.22)	86/0/4 (2.14)
Simba	65/16/9 (1.42)	84/4/2 (2.68)	80/5/5 (2.78)
TensorCore	65/8/17 (1.71)	67/6/17 (1.28)	74/5/11 (1.31)

C.4 Win/Loss Distribution Analysis

Table 12 gives the win/tie/loss distribution against the fair Tuner over the nine accelerator–batch configurations, using the canonical sweep with each case's first completed attempt.

Win rate. CARTA wins at least 72% of the layers in every configuration (lowest: Simba and TensorCore at batch 1) and 99% on Eyeriss at batch 16. The 70 losses concentrate on TensorCore (45; Simba 16, Eyeriss 9) and, by operator, on standard convolutions (28), GEMMs (17), depthwise convolutions (14), and transposed convolutions (11). Their median magnitude is $1.28\times$; the worst is $54\times$ on TensorCore UNet layer 19 at batch 64, where every attempt of the canonical sweep was cut short by API failures (the kept attempt completed 29 evaluator calls); the excess EDP summed over all losses is 0.07% of the suite's total EDP. Decomposing EDP into its factors (Table 13), CARTA's margin comes from both energy and cycles on Eyeriss and Simba, and mostly from energy on TensorCore at batch 16 and 64, where the cycle ratios are 1.05 and 1.07; about half of the losses (39 of 70) lose on both cycles and energy.

C.5 Full-Suite Sample and Time Efficiency

Of the $\sim$200 nominal proposals per layer (40 calls $\times$ 5 decision sets), a median of 164 distinct decision sets reach the evaluator (mean 161.5); these build a median of 92 distinct mappings, of which a mean of 87.9 are admitted to the prompt database. The remainder are proposals whose decision set had already been evaluated, deduplicated before evaluation. Builder rejections are not recorded in the run logs (the agent discards a failed evaluation without logging it), so we report no rejection rate.

We next compare CARTA and the Soter RL Tuner under a common unit of search effort: one evaluator call. The Soter Tuner trains a Transformer policy for each layer using 10 epochs with 32 policy samples per epoch, i.e., 320 evaluated mappings per layer. CARTA's counter increments once per successful evaluation of a new decision set; it omits the single evaluation of the builder's initial

Table 13: Energy and cycle decomposition: geomean ratios fair Tuner/CARTA (> 1 favors CARTA) per configuration, from the stored canonical results (the first-completed-attempt correction changes EDP in 7 cases; cycles and energy of the replaced attempts were not logged).

	Batch 1			Batch 16			Batch 64		
Accelerator	EDP	Cycles	Energy	EDP	Cycles	Energy	EDP	Cycles	Energy
Eyeriss	1.57	1.18	1.33	2.19	1.47	1.49	2.21	1.46	1.51
Simba	1.46	1.28	1.14	2.68	1.99	1.35	2.78	2.11	1.32
TensorCore	1.71	1.50	1.14	1.29	1.05	1.23	1.36	1.07	1.27

Table 14: Full-suite evaluator efficiency on the 270-case batch-16 suite. Soter Tuner uses 320 evaluated mappings per layer. *Eval. calls* are CARTA's counted Timeloop calls on distinct decision sets (add one for the initial mapping), the unit the Tuner's 320 and the controls' budgets count; *DB ins.* are distinct mappings admitted to CARTA's prompt database, which a naive counter conflates with evaluator calls. *Red.* is the reduction in evaluator calls relative to 320. *Ratio* is fair Tuner EDP / CARTA EDP (fair Tuner, identical problems). The Qwen3.5 evaluator-call mean is over the 269 of 270 cases whose run log records the counter (the Simba/BERT-Large layer-0 log is missing).

Backend	**Cases**	**Eval. calls**	**DB ins.**	**Soter**	**Red.**	**Ratio**	**Wins**
DeepSeek API (traced run)	270	161.5	87.9	320	49.5%	1.85×	238/270
GPT-4o-mini	270	176.6	92.1	320	44.8%	1.59×	219/270
Qwen3.5-35B local	270	158.7	78.0	320	50.4%	1.70×	236/270

mapping and the final replay of the best mapping, so CARTA's search evaluator calls are the counter plus one.

CARTA issues 45–50% fewer evaluator calls than the Soter Tuner (161.5, 176.6 and 158.7 per layer for DeepSeek, GPT-4o-mini and Qwen3.5, against 320; the Qwen3.5 mean is over the 269 logged cases) while reaching 1.59–$1.85\times$ lower EDP than the fair Tuner. The traces record progress only when a mapping enters the prompt database, not at every evaluator call, so we do not report anytime or early-stopping curves for the agent; the controls' best-so-far curves in evaluator units are in Figure 7.

The wall-clock result does not favour CARTA: on the full 270-case trace it uses fewer evaluator calls and reaches lower EDP than the Tuner, but it is slower in wall-clock time because LLM serving/API latency dominates the search loop. DeepSeek is the fastest measured CARTA backend at 279s per layer, or $3.34\times$ the traced latency-directed Tuner time, while producing $1.85\times$ lower EDP than the fair Tuner with 49.5% fewer evaluator calls; GPT-4o-mini and the local Qwen service are slower still, though they too reach lower EDP than the Tuner. The local Qwen timing is measured with vLLM 0.18.0 on two NVIDIA H100 80GB HBM3 GPUs using tensor parallelism of 2, bfloat16 weights, a 16K-token context limit, prefix caching, and the OpenAI-compatible API endpoint. Thus CARTA's efficiency claim is in evaluator calls; wall-clock efficiency depends on the serving backend, batching, and API latency, and these timings come from different harnesses rather than a controlled quality–time frontier.

Because all 270 layer cases are independent, both Soter and CARTA scale nearly linearly with worker count in the trace-driven simulation of Table 16: DeepSeek-backed CARTA drops from 20.93h serial to 0.68h at 32 workers, but the relative wall-clock ordering is unchanged because per-case LLM latency remains the dominant cost.

Table 17 makes the tradeoff explicit: CARTA delivers $2.9\times$–$3.7\times$ higher quality per evaluator call than the fair Tuner (DeepSeek, GPT-4o-mini and Qwen3.5; the Qwen3.5 evaluator-call mean is over the 269 of 270 cases with a run log), but lower quality per wall-clock second under the measured API/local-serving configurations. The per-call figure measures the Tuner's inefficiency as much as CARTA's efficiency: random search, with no prior, already has lower EDP than the fair Tuner after 32 evaluator calls, and random search at 2,000 evaluator calls, which an uncontended $\sim$0.1 s evaluator runs within CARTA's wall-clock, reaches parity with the pooled runs ("Equal-wall-clock comparison"). Across backend–accelerator pairs, the reduction in evaluator calls ranges from 44.6% (GPT-4o-mini on TensorCore) to 55.6% (Qwen3.5 on Eyeriss); DeepSeek and GPT-4o-mini alone span 44.6%–49.6%. The fair Tuner/CARTA ratio stays above $1.27\times$ on every DeepSeek and Qwen3.5 accelerator slice, while GPT-4o-mini ties the fair Tuner on TensorCore ($1.00\times$), the hardest architecture for the smaller cloud backend.

Table 15: Full-suite wall-clock time on the 270-case batch-16 suite. The Soter row is a traced rerun of the latency-directed Tuner (same algorithm and 320 evaluator calls) and includes policy training, candidate evaluation, and final map selection; the fair Tuner rerun ran under a capped parallel harness, so its times are not comparable. CARTA rows include LLM inference/API latency, parsing, mapping construction, and analytical evaluation. *Spd.* is fair Tuner EDP / CARTA EDP. "Ratio" is the mean over the 270 cases of the per-case CARTA time / Tuner time (the ratio of the mean times is 2.80/8.84/4.97×); the 3.34–10.88× figures in this appendix are this per-case mean.

Method	Eval. calls	Time	Ratio	Spd.
Latency-directed Tuner	320.0	99.6s	1.00×	1.00×
CARTA + DeepSeek	161.5	279s	3.34×	1.85×
CARTA + GPT-4o-mini	176.6	880s	10.88×	1.59×
CARTA + Qwen local	158.7	495s	6.06×	1.70×

Table 16: Trace-driven parallel scheduling simulation on the 270-case suite. Each case is an independent job with its measured wall-clock time; jobs are assigned to identical workers by longest-processing-time-first list scheduling. Entries are estimated suite makespan in hours.

Method	1w	4w	16w	32w
Latency-directed Tuner	7.47	1.87	0.47	0.24
CARTA + DeepSeek	20.93	5.25	1.31	0.68
CARTA + GPT-4o-mini	66.02	16.59	4.16	2.17
CARTA + Qwen local	37.09	9.30	2.33	1.21

C.6 Token and API-Cost Efficiency

Following recent LLM code-generation benchmarks, which report the token and dollar budgets of agentic workflows alongside correctness and performance (Ouyang et al., 2025; Li et al., 2025; Wang et al., 2025), we instrumented CARTA's LLM client to record prompt and completion tokens for each call. The primary token-efficiency experiments are full 270-case batch-16 runs with DeepSeek-Chat, GPT-4o-mini, and a locally served Qwen3.5-35B-A3B backend, covering the complete layer-selection suite (90 layers × 3 accelerators). The local Qwen backend is served by vLLM 0.18.0 with tensor parallelism of 2 on two NVIDIA H100 80GB HBM3 GPUs, bfloat16 precision, a 16K-token maximum context length, prefix caching enabled, and an OpenAI-compatible endpoint. We additionally run a 12-case multi-model calibration subset (4 workloads × 3 accelerators) to compare other backends under the same search loop. Table 18 reports the mean per-layer token count, API cost, and quality relative to the fair Tuner. For cloud APIs, cost is computed from the input/output token prices encoded in our analysis artifact; local Qwen rows report API cost as "local" and should be interpreted as a separate deployment mode whose cost is GPU time and serving infrastructure.

Full-suite token efficiency. The full DeepSeek-Chat run completes all 270 batch-16 cases with the 40-call per-layer budget. It consumes 109K input tokens and 40K output tokens per layer on average (149K total tokens/layer), for 40.3M total tokens and $7.14 total API cost across the suite. Its EDP is 1.85× lower than the fair Tuner's (geomean), with 238/270 wins. By accelerator, the ratios are 2.26× on Eyeriss, 2.20× on Simba, and 1.28× on TensorCore. The full GPT-4o-mini run uses a similar input-token budget and reaches a 1.59× ratio with 219/270 wins at $11.58 total API cost. The full local Qwen3.5-35B-A3B run consumes more tokens (201K total tokens/layer) but still reaches a 1.70× ratio with 236/270 wins, so the win over the fair Tuner is not confined to a few sampled layers or to one proprietary model. These token-efficiency comparisons are against the Tuner only; against structured search the pinned Qwen3-14B backend loses (0.92×, "A pinned open-weight backend").

Cost-quality tradeoff. On the full 270-case suite, DeepSeek-Chat costs $0.0264 per layer, or roughly $21.4 for an 810-layer sweep under the recorded price schedule. GPT-4o-mini costs $0.0429 per layer on the same full suite and only ties the fair Tuner on TensorCore (1.00×), making DeepSeek-Chat the better measured cloud backend in both cost and quality. On the 12-case calibration subset, the traced DeepSeek run remains the best measured API cost-quality point: a 2.37× ratio at $0.0247 per layer. Gemini Flash-Lite also wins all 12 subset cases but costs 1.30× more per layer and reaches a lower ratio (2.15×). GPT-4o-mini consumes a similar number of input tokens but has higher output-token pricing in our artifact, yielding the highest calibration-subset cost among the three measured API backends.

Table 17: Quality-normalized efficiency on the full 270-case trace. "Q/sample" is the geomean fair Tuner/CARTA EDP ratio per evaluator call (CARTA's 161.5, 176.6 or 158.7 against the Tuner's 320); dividing by prompt-database insertions instead would overstate efficiency. "Q/sec" is the same ratio per wall-clock second, using the traced Tuner time. The comparison is with the Tuner only: random search passes the fair Tuner between 16 and 32 evaluator calls, and structured search reaches parity with the pooled DeepSeek runs at 500 (Sec. 5), so these columns say nothing about CARTA's efficiency relative to search.

Method	Speedup	Q/sample	Q/sec
Latency-directed Tuner	1.00×	1.00×	1.00×
CARTA + DeepSeek	1.85×	3.67×	0.66×
CARTA + GPT-4o-mini	1.59×	2.89×	0.18×
CARTA + Qwen local	1.70×	3.42×	0.34×

Table 18: Token and API-cost efficiency. The first three rows are full 270-case batch-16 experiments; the remaining rows are a 12-case multi-model calibration subset. All rows use the same CARTA algorithm and a 40-call per-layer budget; "Ratio" is fair Tuner EDP / CARTA EDP. The DeepSeek calibration row is the full-suite traced run restricted to the same 12 cases. Cost is API billing only; local Qwen backends are included to show quality under local serving, not to claim zero total cost of ownership.

LLM backend	Scope	Layers	Input tok.	Output tok.	API $/layer	Ratio	Wins
DeepSeek-Chat	full	270	109,496	39,689	0.0264	1.85×	238/270
GPT-4o-mini	full	270	107,587	44,589	0.0429	1.59×	219/270
Qwen3.5-35B-A3B local	full	270	129,296	71,767	local	1.70×	236/270
DeepSeek-Chat (traced run)	calib.	12	106,668	34,722	0.0247	2.37×	12/12
Gemini 2.5 Flash-Lite	calib.	12	119,518	50,292	0.0321	2.15×	12/12
GPT-4o-mini	calib.	12	106,932	42,156	0.0413	2.00×	11/12
Qwen3.5-9B local	calib.	12	127,289	65,347	local	1.89×	11/12
Qwen3.5-122B-A10B-FP8 local	calib.	12	127,709	71,348	local	2.23×	12/12

Token budget interpretation. CARTA spends most tokens on structured architecture, layer, and feedback context rather than free-form chain-of-thought. The full cloud runs average 149K–152K total tokens/layer; the full local Qwen run averages 201K total tokens/layer. These token budgets buy a 40-call closed-loop search and replace the Tuner's per-task RL training loop; the local Qwen runs consume more completion tokens but still outperform the Tuner, so the win over the Tuner is not tied to a single proprietary model or pricing regime.

C.7 Mapping-Level Case Study: A Win and a Loss

To illustrate what CARTA's mappings do differently, we re-ran the agent from scratch on four representative layers and froze the resulting mappings; this is an illustration, not a test of memorization (the contamination control is reported earlier in App. C). The win/loss outcomes reproduce the headline numbers up to the run-to-run variance documented in Sec. 5.3 (median layer ratio 1.02; a minority of first-layer cases vary by more than 2×). Table 19 shows the buffer-level decomposition for the largest win (Simba/DenseNet-161 L0, 9.4×) and a representative loss (Eyeriss/UNet L10, 4.5× in Tuner's favor).

Why CARTA wins (DenseNet-161 L0). The Tuner allocates more PEs (1024 vs 768) but loses by 9.4×. It spreads spatial parallelism over the output-pixel dimensions ($P\times Q$), so each PE computes a 1×1 output tile. CARTA instead spreads it over (i) batch N=16 at the GlobalBuffer, so the 16 batch items share—multicast—the same weights, and (ii) K=16 output channels at the AccumulationBuffer, so each PE accumulates its own outputs locally. The Tuner also streams four dimensions (P, C, K, N) from DRAM against CARTA's two (P, Q); in the Timeloop statistics of the two frozen mappings the Tuner's mapping makes 4.8× the DRAM scalar accesses (106.5M vs. 22.1M) and spends 4.8× the DRAM energy. We read this off the two mappings for this layer; we have not verified that the same pattern explains CARTA's other wins.

Why CARTA loses (UNet L10). The Eyeriss UNet layer-10 has C=1024, K=512 — both very large reduction-dominated dimensions. The Tuner partitions C across four loop levels (C=16 at

Table 19: Mapping-level decomposition of one win (top) and one loss (bottom). "Spatial" lists per-buffer parallel factors (product over buffers = PE utilization). "Temporal" lists per-buffer loop tiles. In this win, CARTA allocates spatial parallelism to dimensions that maximize data reuse rather than using more PEs (col. "PE used"). The loss case shows a mapping that leaves the $C{=}1024$ reduction dimension untiled across levels. Tuner rows are the latency-directed Tuner's mappings from the rerun used for this illustration; against the fair Tuner the two ratios are $13.2\times$ and $0.11\times$.

Buffer	Side	Spatial	Temporal	PE used	DRAM dims
Top-win: Simba / DenseNet-161 layer 0 — CARTA EDP = 2.625×10^{11} *vs Tuner* 2.477×10^{12} *(9.44×)*					
DRAM	CARTA	—	P=7, Q=56	768	2
DRAM	Tuner	—	P=14, C=3, K=4, N=8		4
GlobalBuffer	CARTA	N=16	—	1024	—
GlobalBuffer	Tuner	K=8, N=2	—		—
AccumulationBuf	CARTA	C=3, K=16	R=7, S=7, P=8, K=6		—
AccumulationBuf	Tuner	P=4, Q=16	R=7, S=7, Q=7, K=3		—
Loss: Eyeriss / UNet layer 10 ($C{=}1024$, $K{=}512$*) — CARTA EDP* = 6.76×10^{14} *vs Tuner* 1.50×10^{14} *(0.22×)*					
DRAM	CARTA	—	S=2, P=14, Q=28	256	3
DRAM	Tuner	—	P=7, C=16, K=2, N=16		4
GlobalBuffer	CARTA	K=16	R=2, P=2, **C=1024**, N=2	256	—
GlobalBuffer	Tuner	C=4, K=4	K=8		—
DummyBuffer	CARTA	K=16 (unused)	—		—
DummyBuffer	Tuner	R=2, S=2, P=4	Q=28, C=2, K=8		—

DRAM $\times$ $C{=}4$ in GlobalBuffer spatial $\times$ $C{=}2$ in DummyBuffer $\times$ $C{=}8$ in the PsumRegFile, a register level Table 19 omits), achieving fine-grained reuse. CARTA leaves $C{=}1024$ as a single innermost temporal loop inside the GlobalBuffer, producing a small tile that has to re-stream from DRAM 1024 times. In this layer the agent did not propose a multi-level split of the single large reduction dimension; one layer does not establish how general that failure is, and it is not evidence about contamination.

D Deployment Cost Detail

D.1 Deployment Cost and Token Efficiency

CARTA spends exactly 40 LLM API calls per layer plus a median of 164 analytical-model evaluator calls; at the 0.1–0.18 s per call measured uncontended and in the Tuner traces, the evaluator time is roughly 16–30 s, and about 80 s under our 20-worker harness (App. C). To price the API calls in dollars, we instrumented the DeepSeek client to record per-call `prompt_tokens` and `completion_tokens` and re-ran the agent on a 12-layer subset spanning all three accelerators and four representative workloads (Table 20).

A layer costs $0.0254 even though every call re-sends the static architecture description, and the full 810-layer benchmark suite costs $\sim$$21 at current DeepSeek pricing; because Table 20 reports the token counts, the estimate can be recomputed under any future price change.

Per-phase cost breakdown. Call counts and prompt lengths, not completion lengths, set each phase's share of the 40-call budget's cost (Table 21). All three phases ask for five complete decision objects per call, so their completions are of similar length (889, 967 and 904 output tokens per call for exploration, refinement and polishing), and the phase shares follow from the call counts (18/12/10) and the prompt lengths. Exploration and refinement prompts average 2,930 and 3,266 input tokens per call because both carry the long system prompt and, from the second exploration call on, the top-5 history; the polishing prompt averages 1,623 tokens because it replaces the system prompt with a short instruction and shows a single parent. Polishing is therefore the cheapest phase ($0.0048, 18.9% of cost), exploration takes 46.7% of the cost against 45% of the calls, and refinement 34.4%.

Prompt-caching opportunity. Much of each prompt could be served from a prompt cache. The architecture description is byte-identical across all 40 calls within a layer, and the system prompt across the exploration and refinement calls (polishing replaces it with a short instruction, App. G). Taking the floor of `prompt_tokens` across calls as the static prefix, 59.8% of input tokens are static (mean 1,623-token prefix per call). This floor is a heuristic estimate of the cacheable share

Table 20: Measured per-layer API cost of CARTA (DeepSeek-Chat/V4-Flash, 40 calls, batch 16). Tokens are summed over all 40 calls in the three-phase search. Dollar cost uses current DeepSeek cache-miss pricing: USD 0.14 per million input tokens and USD 0.28 per million output tokens. Means are over 12 layers (4 workloads $\times$ 3 accelerators). TensorCore has fewer buffer levels (3 vs Simba/Eyeriss's 5, excluding DRAM), shortening the LLM's JSON output by $\sim$30% and reducing cost by $\sim$15% relative to the Eyeriss/Simba mean.

Accelerator	In tokens	Out tokens	Cost / layer
Eyeriss	105,063	39,340	\$0.0257
Simba	112,719	42,671	\$0.0277
TensorCore	106,690	27,907	\$0.0228
Mean (n=12)	**108,158**	**36,640**	**\$0.0254**

Table 21: Per-phase token and dollar breakdown (mean of 12 layers, DeepSeek-Chat pricing).

Phase	Calls	In tok	Out tok	Cost	Share
Explore	18	52,743	16,000	\$0.0119	46.7%
Refine	12	39,187	11,602	\$0.0087	34.4%
Polish	10	16,228	9,037	\$0.0048	18.9%
Total	**40**	**108,158**	**36,640**	**\$0.0254**	**100%**

rather than a bound: in every layer it is the first polishing prompt, which already contains layer-specific content (the current best mapping and its energy breakdown), so it over-counts the cacheable share of the 10 polishing calls and under-counts the longer static prefix (system prompt plus architecture description) shared by the 30 exploration and refinement calls. At DeepSeek's prompt-cache pricing (\$0.0028 per million cached-prefix tokens vs. \$0.14 uncached) the per-layer cost falls from \$0.0254 to \$0.0165, a 35.1% saving, and the full 810-layer suite from \$21 to \$13. The cache saves only input cost: about 40% of the per-layer bill is the completion (mapping JSON), which caching does not address.

Backend cost-quality Pareto. To check that the cost result is not specific to one provider, we re-ran the same 12-layer subset with low-cost API alternatives and locally served Qwen3.5 models under the identical search code. Table 22 reports token usage and mapping quality against the fair Tuner on these 12 cases.

All six backends beat the fair Tuner on this 12-layer subset, by 1.89–2.37$\times$. The API backends cost under five cents per layer; the local Qwen3.5 runs remove cloud token charges and shift the cost to GPU availability. The quality ordering does not strictly follow model size or price: Qwen3.5-35B-A3B matches the 122B-A10B-FP8 ratio and is close to DeepSeek-Chat, which makes it the most attractive local backend in our experiments. This is not a small-subset artifact: on the full 270-case batch-16 suite (Table 28) Qwen3.5-35B-A3B still reaches a 1.70$\times$ ratio with 236/270 wins against the fair Tuner. CARTA's margin over the fair Tuner therefore needs neither an expensive frontier model nor a proprietary API.

Latency and infrastructure. Wall-clock time is dominated by the LLM API round-trip. The traced DeepSeek run averages 279 s per layer, about 7 s per call including parsing and the $\sim$30 s of evaluator time, against 99.6 s for the latency-directed Tuner including its per-task policy training (Table 15). Parallel workers shorten the suite makespan (Table 16) but leave the per-layer ordering unchanged. The Tuner pays no API cost but carries the implicit cost of maintaining the RL pipeline (Transformer policy, per-task policy-gradient training, hyperparameter tuning) for each new accelerator class.

Table 22: Cost-quality Pareto across API and local LLM backends on the same 12-layer subset. "Ratio" is geomean fair-Tuner EDP / CARTA EDP (fair Tuner, identical problems); wins count cases where the LLM result beats that Tuner. The DeepSeek row is the full-suite traced run restricted to these 12 cases. Cost uses list prices per million input/output tokens: DeepSeek USD 0.14/0.28, Gemini 2.5 Flash-Lite USD 0.10/0.40, GPT-4o-mini USD 0.15/0.60.

Model	In/Out tok	Cost	Ratio	Wins
DeepSeek-Chat (traced run)	107K / 34.7K	$0.0247	2.37×	12/12
Gemini 2.5 Flash-Lite	120K / 50.3K	$0.0321	2.15×	12/12
GPT-4o-mini	107K / 42.2K	$0.0413	2.00×	11/12
Qwen3.5-9B local	127K / 65.3K	$0†	1.89×	11/12
Qwen3.5-35B-A3B local	128K / 67.2K	$0†	2.23×	11/12
Qwen3.5-122B-A10B-FP8 local	128K / 71.3K	$0†	2.23×	12/12

†Local rows report cloud API token cost as zero; GPU amortization, power, and serving overhead are not included.

A new accelerator needs a new description and, on the evidence of our ports, a strategy text transcribed from random-search winners on development layers (3,000 diagnostic evaluator calls plus expert reading per port), but no retraining. We do not claim that CARTA's advantage grows with design-space complexity: the pre-registered fanout sweep (App. C) falsified that prediction in the opposite direction.

E Per-Task RL Training Limitations

E.1 Limitations of Per-Task RL Training

Soter's RL Tuner is a Transformer policy, trained by policy gradient, that emits a mapping one loop dimension per decoding step, predicting its loop order, tile exponents and spatial factor buffer level by buffer level. This design has four limitations. They are properties of the design, not measured causes of the Tuner's gap: our experiments do not test which of them, if any, explains it.

1. **Cold start.** For each new (layer, accelerator) pair the Transformer starts from random weights and requires 10 epochs × 32 candidates = 320 evaluator calls to converge. Nothing carries over between problems, not even between two 3×3 convolutions that differ only in channel count.

2. **Architecture coupling.** The policy's weights encode the accelerator it was trained on. A policy trained on Simba's 5-level weight-stationary hierarchy holds none of the knowledge needed for Eyeriss's per-tensor register files or TensorCore's 3-level GPU hierarchy, so N accelerators need N independent policies.

3. **Sequential decisions.** The Transformer fixes one parameter at a time in a fixed order, each conditioned on those already sampled, so an early choice cannot be revised once later choices expose its interaction with them. Spatial parallelism and tiling interact this way: spatial $K = 8$ favors large output-channel tiles, whereas spatial $P = 8$ favors large spatial tiles.

4. **Scalar reward.** The Tuner sees only the final EDP of each mapping, a single number that says nothing about *why* the mapping is good or bad. It cannot tell "DRAM energy dominates because tiling is insufficient" from "the accumulation buffer is underutilized because the spatial allocation is poor." We have not shown that richer feedback helps: in our mechanism ablation, replacing CARTA's per-level energy breakdown by scalar EDP shows no detectable harm ($0.94\times$, paired t-test p=0.42; Table 10).

The search controls of App. C place the Tuner's quality in context. Random search, which learns nothing, passes the fair Tuner between 16 and 32 evaluator calls on the main suite, whereas on NVDLALike random search, SA and the GA at 500 evaluator calls only tie it. A training-free proposer removes this per-task cost, at the price of per-layer search, wall-clock time and, for a new accelerator, strategy text developed from diagnostic evaluations (Sec. 5.4).

F CANDIDATE MECHANISMS FOR AN LLM PRIOR

F.1 FOUR CANDIDATE MECHANISMS AND WHAT THE CONTROLS SAY

Four mechanisms could explain why an LLM proposer helps; our experiments establish none. Two are contradicted below and two were not tested. What the controls do support rests on the strategy text (the ablation of Sec. 5.3, the strategy-seeded annealer, and the ports of App. C).

1. **Joint proposals.** The LLM emits every mapping decision in one response, which might capture cross-parameter interactions that a sequential policy handles poorly. *Not tested: no experiment separates joint proposal from the builder or the strategy text.*
2. **Causal feedback interpretation.** Because the LLM sees per-level energy breakdowns while the RL Tuner sees only a scalar reward, it could target the memory levels that dominate energy. *Not supported: the mechanism ablation of App. C (Table 10) finds no measurable effect of replacing the breakdown by scalar EDP.*
3. **Cross-architecture transfer.** Mapping knowledge might transfer across accelerators through the model itself; RL policies are trained per task. *Not supported. On a held-out accelerator with no hand-written strategy text, the agent loses to every matched-budget search (*0.74–$0.82\times$*) and to the fair Tuner (*$0.86\times$*); the latency-directed Tuner yields an apparent* $1.76\times$ *win that reverses once the objective and problems are aligned. About 30 lines of strategy text, transcribed from random-search winners on 10 development layers, restore a win over the fair Tuner and over knowledge-free search on the 20 GemminiLike test layers, and a strategy-seeded annealer given the same text recovers most of that win (App. C). On NVDLALike the ported text gives DeepSeek wins over knowledge-free search but only a tie with the fair Tuner (*$1.02\times$*, CI* $[0.94, 1.10]$*), and the registered Qwen3-14B arm still loses to the fair Tuner (*$0.67\times$*) and to every search control. What transfers is the text, not a property of the model.*
4. **Strategy-level search.** Proposals phrased as strategies rather than individual parameters might act as a hierarchical search. *Not tested.*

G ALGORITHM AND PER-PHASE PROMPT SCHEMA

Call and evaluator-call counts. Refine can issue fewer than its nominal N_2 calls. It takes the bases in order, best first, spending $\max(2, \lfloor N_2/|\text{bases}| \rfloor)$ consecutive calls on each, so it issues $|\text{bases}| \cdot \max(2, \lfloor N_2/|\text{bases}| \rfloor)$ calls: 12 at $N{=}40$ with three bases, but only 6 when a smaller budget's $N_2{=}8$ meets three bases. The scripted hints cover only the first 12 Explore calls, so the radical-change prompt can fire only in the later ones. Each call yields at most five candidates, so the per-phase caps of 90, 60 and 50 evaluator calls are upper bounds. In the traced DeepSeek batch-16 run (270 cases) the per-phase medians are 90, 59 and 15 evaluator calls (cumulative 90, 149, 164); the per-layer total has median 164, mean 161.5 and range 26–198, and no case reaches 200.

Duplicate proposals. In Explore and Refine, a proposal that repeats an already-evaluated decision set is randomly perturbed once: with probability 0.5 the spatial allocation of one level is redrawn from its first 30 full-utilization options, and with probability 0.7 two dimensions of one loop-priority list are swapped; if the result still repeats, the proposal is skipped. Polish does not perturb repeats. These random edits are part of the complete system our experiments evaluate; we do not isolate their contribution from the LLM's.

Mapping builder. The builder (`utils/mapping_builder.py`) turns a decision set into a Timeloop mapping deterministically, rounding rather than rejecting what it cannot honour. A decision set names a spatial allocation for every spatial level (dimension factors whose product must equal that level's fanout), a loop-priority list for every buffer level, and optionally tile caps per level and dimension; the schema appears in the verbatim prompt of App. A. It first assigns spatial factors, innermost level first, snapping each requested factor to the largest divisor of the remaining dimension size that also divides the level's unused fanout, and fills any leftover fanout greedily. It then assigns temporal factors, innermost level first, walking each level's priority list and giving each dimension the largest divisor whose tile fits that level's capacity and every outer level's; a tile cap is

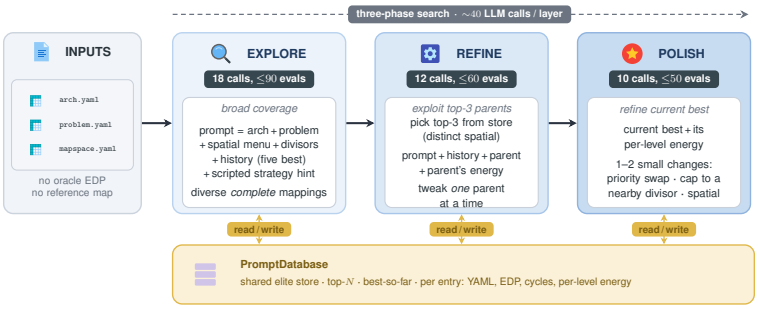

Figure 12: CARTA three-phase search pipeline. Phase 1 *Explore* proposes diverse complete decisions, guided by scripted strategy hints and the five best results so far, to seed the PromptDatabase; Phase 2 *Refine* takes up to three top parents with distinct spatial allocations and perturbs each in turn; Phase 3 *Polish* reads only the current best and adds its energy breakdown to the prompt, asking the LLM for one or two small changes. All three phases read and write the shared PromptDatabase, so later phases see the elites accumulated by earlier ones. The phase badges are caps, not counts: 18, 12 and 10 LLM calls of up to five candidates each, at most 200 evaluator calls per layer. Repeated decision sets are skipped and candidates the builder or Timeloop rejects are not counted, so the traced DeepSeek batch-16 run counts a median of 164 evaluator calls per layer (mean 161.5, range 26–198; per-phase medians 90, 59 and 15; none of the 270 cases reaches 200).

Table 23: User-prompt contents per search phase. Explore and Refine share one system prompt (general mapping rules, the decision schema and the per-accelerator strategy text); Polish replaces it with a short fine-tuning instruction. "History" is the five best PromptDatabase entries, each printed with its decisions, EDP, cycles, utilization and an energy line naming the components above 10% (bottleneck marker above 40%); "parent" is the configuration the LLM is asked to modify (each of up to three top configurations with distinct spatial allocations in Refine, the current best in Polish). The scripted strategy hint is a per-call task line, distinct from the strategy text carried by the system prompt. The scalar-feedback ablation removes both the energy line and the full breakdown, so it changes Explore as well as Refine and Polish.

Prompt block	**Explore**	**Refine**	**Polish**
Arch description (levels, caps, sp_cstr)	✓	✓	✓
Problem dims + valid divisors	✓	✓	✓
Spatial menu (PE factorizations)	✓	✓	✓
Scripted strategy hint (12; calls 1–12 only)	✓		
History: five best elites, with energy line	✓	✓	
Parent config to perturb		✓	✓
Full energy breakdown (parent in Refine, best in Polish)		✓	✓
Nearby divisors for the parent's caps			✓
Change directive (1–3 changes in Refine, 1–2 in Polish)		✓	✓

honoured when it fits and otherwise reduced to the largest fitting divisor; and two small heuristics restrict which dimensions tiny single-datatype buffers and bypass levels tile. A final pass shrinks temporal factors wherever a cross-level tile still exceeds a buffer, and DRAM absorbs the leftover factors. Divisibility therefore holds by construction, and capacity and fanout by these checks; after rounding, the mapping evaluated can differ from the one the LLM asked for.

Algorithm 1 CARTA Three-Phase Search

Require: Layer spec L, architecture spec A, budget N=40 LLM calls
Ensure: Best mapping m^* with minimum EDP

$m^* \leftarrow$ builder's initial mapping; EDP$^* \leftarrow$ AnalyticalModel(m^*, L, A) `// not counted in the budget`
hints $\leftarrow$ {12 scripted strategy hints} `// the code lists 14; only the first 12 are ever sent`
$N_1, N_2, N_3 \leftarrow \max(10, \lfloor 0.45N \rfloor), \max(8, \lfloor 0.30N \rfloor), \max(7, \lfloor 0.25N \rfloor)$ `// 18, 12, 10 at` N=40
`// Phase 1: Exploration`
for $i = 1$ **to** N_1 **do**
  if $i \leq 12$ **then**
    $s_i \leftarrow$ hints$[i]$
  else if no improvement for more than 8 calls **then**
    $s_i \leftarrow$ radical-change prompt
  else
    $s_i \leftarrow$ generic diversity prompt
  end if
  $\{c_1, \ldots, c_k\} \leftarrow$ LLM(L, A, s_i, five best so far), $k \leq 5$
  for each candidate c_j whose decision set is new ("Duplicate proposals" below) **do**
    $m_j \leftarrow$ MappingBuilder(c_j, A); EDP$_j \leftarrow$ AnalyticalModel(m_j, L, A) `// counted only if both succeed`
    if EDP$_j <$ EDP* **then**
      $m^* \leftarrow m_j$; EDP$^* \leftarrow$ EDP$_j$
    end if
  end for
end for
`// Phase 2: Refinement`
bases $\leftarrow$ up to three best configurations with distinct spatial allocations
for each base $b \in$ bases, best first **do**
  for $i = 1$ **to** $\max(2, \lfloor N_2 / |\text{bases}| \rfloor)$ **do**
    feedback $\leftarrow$ EnergyBreakdown(b)
    $\{c_1, \ldots, c_k\} \leftarrow$ LLM(L, A, b, feedback, five best so far)
    Evaluate and update m^* as above
  end for
end for
`// Phase 3: Polishing (own system prompt)`
for $i = 1$ **to** N_3 **do**
  feedback $\leftarrow$ EnergyBreakdown(m^*)
  $\{c_1, \ldots, c_k\} \leftarrow$ LLM(L, A, m^*, feedback, small-change directive)
  Evaluate and update m^* as above (repeats are skipped, not perturbed)
end for
return m^*

H PARADIGM COMPARISON TABLE

Table 24: Paradigm comparison. Neither CARTA nor the search controls train per task; CARTA differs from them in proposing mappings with a frozen LLM, and it still searches. For the named compiler frameworks, the "Per-task training?" and "Per-task evaluator calls" columns describe our template-constrained re-implementations, not the original tools: the arms we run fit no learned model, whereas the original AutoTVM and Ansor fit a per-task learned cost model and the original FlexTensor adds Q-learning; likewise, our CoSA-labeled arm does not implement CoSA's mixed-integer program. "Per-task evaluator calls" counts the mappings each method evaluates per (layer, accelerator) in our setup (Sec. 4); "n/a" marks a framework we did not run (AMOS), and "n/a (run, not reported)" a Mind Mappings-style arm we ran but whose results no table in this paper reports. "New arch" and "New layer" give the work needed to support a new accelerator or a new DNN operation.

Framework	Per-task training?	Per-task evaluator calls	New arch adaptation	New layer adaptation
AutoTVM (Chen et al., 2018b)	None (template + SA)	500	Retune	Retune
FlexTensor (Zheng et al., 2020b)	None (SA, 5 restarts)	500	Retune	Retune
Ansor (Zheng et al., 2020a)	None (evolutionary)	500	Retune	Retune
CoSA (Huang et al., 2021) (heuristic arm)	None (greedy + hill climbing)	500	Retune	Retune
AMOS (Zheng et al., 2022)	Search	n/a	Re-abstract	Re-search
Mind Mappings (Hegde et al., 2021)	Train surrogate	n/a (run, not reported)	Retrain	Retrain
Soter Tuner (Wang et al., 2024)	Train RL policy	320	Retrain	Retrain
Unconstrained random / SA / GA / TPE / GP-BO (controls)	None	200 (curves to 500)	None	None
Strategy-seeded annealer (structured search control)	None	200 (curves to 500)	Edit strategy text + transcribe templates	None
CARTA	**None**	**median 164 + 1 initial (40 LLM calls)**	**Edit prompt**	**Edit prompt**

I TARGET ACCELERATORS DETAIL

Table 25: The three target spatial accelerators as modeled in Timeloop. Each model enforces capacities, spatial fanouts, and which tensors a level keeps or bypasses (a datatype constraint only); it does not fix which dimensions a fanout parallelizes or the loop order. The published dataflows in the last column are therefore not enforced, for CARTA and every baseline alike. Level counts exclude DRAM. The Eyeriss entry is a 256-PE (16×16) Eyeriss-like Timeloop model, whereas the fabricated chip has a $12 \times 14 = 168$-PE array. The Simba entry models one chiplet of the published multi-chip module (MCM): 16 PEs $\times$ 64 MACs $=$ 1024 MACs, which the table and text count as PEs.

Accelerator	**Memory hierarchy**	**Key characteristic**
Eyeriss-like (Chen et al., 2016)	5 levels + DRAM, 256 PEs Separate I/W/O RegFiles Dummy→Global→DRAM	Row-stationary (published; not enforced) Per-tensor buffer banks 16×16 spatial (modeled)
Simba (Shao et al., 2019)	5 levels + DRAM, 1024 PEs Reg→Acc→Weight→ Input→Global→DRAM	Weight-stationary (published; not enforced) MCM (published; one chiplet modeled) 64×16 spatial
TensorCore (NVIDIA, 2018)	3 levels + DRAM, 1024 PEs LRF→RF→SMEM→DRAM	GPU-style hierarchy Large shared memory 256×4 spatial

J LLM BACKEND SENSITIVITY TABLES

What the budget runs show. Every budget from 25 to 80 executed calls beats the fair Tuner on Simba, by 2.12–$2.77\times$, and the margin is not monotone in the budget: the 25- and 29-call single runs (2.13–$2.25\times$) fall between the two 40-call results, the pooled $2.12\times$ and the canonical $2.68\times$, so the spread between runs at a fixed budget is as large as any budget effect. The table supports neither of two further readings. It does not isolate a phase contribution: because the per-phase minimums make every row run all three phases, the 18- and 30-call rows are not exploration-only or no-polish variants, and we draw no phase conclusion from them. Nor does it support a sample-efficiency claim: the stored result files carry no evaluator counts, and the run logs, overwritten on resume, retain per-case counts for only 21–73 of the 90 layers per budget, not all of them attributable to a stored row, so we report none.

LLM model sensitivity. To test whether CARTA's margin depends on one specific LLM, we re-ran the full 270-case batch-16 suite with two further backends under the identical search loop and 40-call budget: GPT-4o-mini (OpenAI API) and a locally served open-weight Qwen3.5-35B-A3B (Table 27; per-layer traces feed Tables 14–15). A Claude Opus arm is excluded for two reasons. First, it ran through the Claude Code CLI rather than an API, on a 90-case subset only (30 layers per accelerator), and the CLI's usage cap refused 802 calls, which the agent logged and skipped, so 27 of its 77 logged cases finished with fewer than 100 evaluator calls (26 with none). Second, the CLI session has read access to the working tree, so its outputs cannot be certified free of contamination from repository artifacts.

Beating the fair Tuner does not hinge on one vendor's model: all three backends win on at least 81% of layers, DeepSeek-Chat is the strongest, and the local open-weight model retains 86% of its geomean ratio. Because every row is a single run, the ordering among backends should be read against the $1.06\times$ overall gap between the canonical DeepSeek run and the pool of three later runs (Section 5).

Local open-weight LLM backend. CARTA's margin over the fair Tuner requires neither a cloud API nor a proprietary model. We served Qwen3.5-35B-A3B behind the same OpenAI-compatible interface and re-ran the full batch-16 paper suite (90 selected layers $\times$ 3 accelerators); the service uses vLLM 0.18.0 on two NVIDIA H100 80GB HBM3 GPUs with tensor parallelism of 2, bfloat16 precision, a 16K-token context limit, prefix caching enabled, and chunked prefill enabled by vLLM. Table 28 breaks the $1.70\times$ geomean fair-Tuner/CARTA EDP ratio (236/270 wins) down by accelerator: the same CARTA search loop beats the fair Tuner on all three accelerator classes, by $2.05\times$ on Eyeriss and $1.84\times$ on Simba, and least on TensorCore ($1.29\times$, 75/90 wins), the weakest architecture for this backend.

Table 26: LLM-call-budget runs (Simba, batch 16, 90 layers), measured against the fair Tuner (EDP objective, identical problems). The agent enforces per-phase minimums of 10/8/7 calls, so nominal budgets of 10, 18 and 30 execute 25, 25 and 29 calls and *every* row runs all three phases. Rows are independent single runs except the pooled row; ratios are fair-Tuner EDP / agent EDP.

Nominal budget	Calls executed (E/R/P)	Fair Tuner / agent	W/T/L (90)
10	25 (10/8/7)	2.25×	79/6/5
18	25 (10/8/7)	2.20×	77/4/9
30	29 (13/9/7)	2.13×	77/3/10
40, canonical run	40 (18/12/10)	2.68×	84/4/2
40, three later runs pooled	40 (18/12/10)	2.12×	80/4/6
80	80 (36/24/20)	2.77×	85/4/1

Table 27: LLM backend sensitivity on the full 270-case batch-16 suite: geomean fair-Tuner EDP / CARTA EDP (fair Tuner on identical problems); wins at the $\pm 1\%$ band. Each row is a single run with the same prompts, budget, and builder.

LLM Backend	Geomean	Wins	Win Rate
DeepSeek-Chat (canonical run, Table 1)	1.97×	240/270	89%
GPT-4o-mini	1.59×	219/270	81%
Qwen3.5-35B-A3B (local, open weights)	1.70×	236/270	87%
Fair Tuner (baseline)	1.0×	—	—

Table 28: Local open-weight Qwen3.5-35B-A3B backend on the batch-16 paper suite (90 selected layers × 3 accelerators); ratios are fair-Tuner EDP / CARTA EDP.

Accelerator	Cases	Ratio	Wins	Win Rate
Eyeriss	90	2.05×	87/90	97%
Simba	90	1.84×	74/90	82%
TensorCore	90	1.29×	75/90	83%
Overall	**270**	**1.70×**	**236/270**	**87%**

K Extended Related Work

Auto-tuning compilers and DSE frameworks. Prior mappers differ in whether their knowledge is templated, searched, or trained per task. The schedule/algorithm separation of Halide (Ragan-Kelley et al., 2013) and TVM (Chen et al., 2018a) underpins the template-based or evolutionary search of AutoTVM (Chen et al., 2018b), FlexTensor (Zheng et al., 2020b), and Ansor (Zheng et al., 2020a); CoSA (Huang et al., 2021) casts mapping as a constrained ILP; Timeloop (Parashar et al., 2019), ZigZag (Mei et al., 2021), and LOMA (Symons et al., 2021) pair analytical evaluation with memory-centric design-space exploration but search heuristically. Multi-objective mappers such as MEDEA (Russo et al., 2022), dMazeRunner (Dave et al., 2019), and GAMMA (Kao & Krishna, 2020)/MAESTRO (Kwon et al., 2020) require problem-specific initialization, and analytical or constraint-based frameworks such as Interstellar (Yang et al., 2020), FLAT (Kao et al., 2023), and HASCO (Xiao et al., 2021) rely on hand-built cost models or templates. Mind Mappings (Hegde et al., 2021) trains a differentiable surrogate; Soter (Wang et al., 2024) pairs an analytical model with a Transformer-based RL tuner; AMOS (Zheng et al., 2022) and LISA (Li et al., 2022b) target spatial accelerators but still require per-architecture data or templates. The learned proposers (Mind Mappings, Soter, LISA) train per task or per architecture; AMOS and the constraint and heuristic mappers avoid training but, like CARTA, search per task. CARTA replaces their hand-built proposer with a frozen LLM guided by a strategy text.

LLMs for code and algorithm discovery. LLMs also serve as optimizers in compiler and hardware work. LLM Compiler (Cummins et al., 2024) pre-trains on LLVM IR and GPT4AIGChip (Fu et al., 2023) generates RTL; AlphaCode (Li et al., 2022a), FunSearch (Romera-Paredes et al., 2024), and AlphaEvolve (Novikov et al., 2025) establish LLMs as optimization engines. CARTA shares this vision but spends a median of 164 evaluator calls in a structured three-phase search with energy-breakdown feedback, rather than evolving populations.

Concurrent data-centric LLM tensor optimization. A concurrent line of work argues the opposite of our thesis: that general-purpose LLMs are insufficient for tensor optimization without domain-specific *post-training*. Step-TP (Liu et al., 2026) builds a grounded, step-level chain-of-thought dataset to fine-tune LLMs for *GPU* tensor-program optimization (loop tiling, fusion, and thread-block binding lowered to TVM TIR). CARTA is complementary and differs on three axes. (i) *Target*: we optimize tensor *mapping* onto spatial-accelerator memory hierarchies (Timeloop), not GPU kernel schedules. (ii) *Method*: we are *training-free*: a frozen pre-trained model, no dataset construction or fine-tuning, whereas Step-TP's contribution is precisely the post-training data. (iii) *Evidence*: rather than assuming general LLMs are brittle, we show a frozen LLM, given a strategy text, already beats Soter's per-task RL Tuner on spatial-accelerator mapping on the three main accelerators and on one of two ports (on NVDLALike DeepSeek ties it and Qwen3-14B loses). The two works take opposite positions (training-free vs. data-centric) on different hardware, and our result bounds how much can be achieved *without* any post-training.