PIPESEARCH: BUYING COMPOSITION LABELS ONLY WHERE THEY CHANGE THE PIPELINE

Anonymous authors
Paper under double-blind review

ABSTRACT

Serving a large language model on a single accelerator asks quantization, pruning, low-rank factorization and cache compression to share one memory and throughput budget, so what is deployed is an ordered composition of methods rather than a single choice. Choosing that composition needs the pairwise interactions of the methods, yet they have to be bought: a pairwise label costs a full compression pass, reversing two methods yields a different model, and some pairings refuse rather than degrade. We propose PipeSearch, which buys a pairwise label only while it can still change the returned pipeline, that is, while the pipelines of its optimistic and pessimistic completions disagree. Decision-relevant acquisition observes the cheapest unobserved pair either completion applies, order-aware prefix search carries which method acted last and keeps a sequence apart from its reverse, and feasibility maintenance excludes a refused pair at expansion time. The key insight is that the observations which reveal the interaction structure are part of the decision rather than a preliminary to it. Across catalogues of four to twelve atoms, PipeSearch returns the pipeline of complete offline labelling while building 0.201 to 0.500 of the $K(K-1)$ ordered pairs, at 75.1 accelerator-hours against 367.4 at K=12. Paying for every candidate pair instead leaves perplexity, zero-shot average, peak memory and decode rate unchanged at both memory budgets, at 15.0 and 3.6 times the search cost. Under a cap of $0.14\times$ the fp16 footprint, where no single method fits, PipeSearch fits Llama-3.1-8B at a predicted 8.79 GB and 9.59 perplexity.

1 INTRODUCTION

Deploying a large language model on a single accelerator asks quantization, pruning, low-rank factorization and cache compression to share one pool of device memory, and none of them alone brings a half-precision backbone within that budget (Frantar et al., 2022; Frantar & Alistarh, 2023; Kwon et al., 2023). What is served is therefore an ordered arrangement of several such methods rather than a single choice. Each method rewrites the model the next one receives. Which methods act, in what order, and at what settings decides the served quality of the deployment and not only its footprint (Gong et al., 2024). The fp16 footprint exceeds the memory cap several times over, and the arrangement has to recover most of the quality each method reaches on its own.

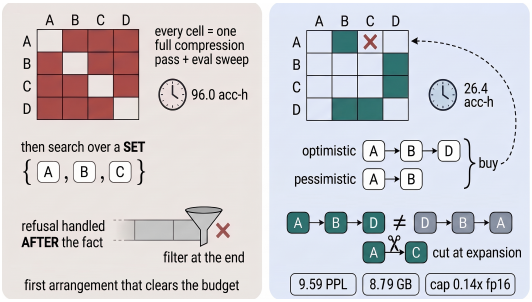

Figure 1: **Left:** a search over a given table labels every pair, searches a set, and filters refusals last. **Right:** PipeSearch buys a label on a pair its completions apply, keeps a sequence apart from its reverse, and cuts a refusal at expansion.

Joint compression methods combine sparsification with quantization, and with a low-rank correction, in one fixed recipe (Guo et al., 2024; Mozaffari et al., 2025). A second line deepens a single method instead: static channel-wise quantization reaches strong operating points (Wang et al., 2025), migrating activation outliers into the weights keeps lower bit widths usable (Xiao et al., 2023), codebook quantization reaches more extreme bit budgets (Egiazarian et al., 2024), and paged caching inside the serving engine reduces the memory that sits outside the weights (Kwon et al.,

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.

2023). Together these lines have moved single-accelerator deployment from whether a model fits to how much served quality survives once it does. Neither line decides which methods act and in what order, and a search that does needs the pairwise interaction structure in hand before it begins: it labels every pair, searches a set and filters refusals last, as the left panel of Figure 1 illustrates.

That structure has to be bought, one full compression pass and one evaluation sweep at a time. A catalogue of K compression methods carries $K(K-1)$ ordered pairs, and reading all of them costs 367.4 accelerator-hours at K=12. Cost is not the only difficulty: applying two methods in the reverse order yields a different model. Some pairs refuse each other outright, and a single refusal deletes a whole region of the arrangement space. A search over a given table nonetheless treats the interaction structure as background knowledge a search may consult, while it is an object the search has to purchase, one compression pass at a time.

To address these three difficulties, we propose PipeSearch, one loop that chooses an ordered pipeline and the observations that entitle it under a single budget. It runs three components, shown in the right panel of Figure 1: decision-relevant acquisition, which compares the pipelines of its optimistic and pessimistic completions and buys the cheapest unobserved pair either completion applies; order-aware prefix search, which carries which method acted last, keeping a sequence apart from its reverse; and feasibility maintenance, which removes a refused pair from expansion once the refusal has been observed. The key insight is that the observations which reveal the interaction structure are part of the decision rather than a preliminary to it: a sequence and the labels that entitle it are chosen under one budget. Our contributions are as follows:

- We identify that the interaction structure of composed compression is purchased rather than given, and its acquisition cost, its order sensitivity and its outright refusals form one decision rather than three that a search may settle separately.
- We propose PipeSearch, an order-aware prefix search that pays for a pairwise label only while its two extreme completions disagree on the returned sequence, keeps a sequence apart from its reverse, and excludes a refused pair at expansion time.
- Across catalogue sizes from $K = 4$ to $K = 12$, PipeSearch returns the pipeline of complete offline labelling while paying for 0.201 to 0.500 of the $K(K-1)$ pairs, and in a four-arm ablation it matches paying for every candidate pair at 6.4 and 26.4 accelerator-hours against 96.0, at the two memory budgets it is read on.

2 MOTIVATION

To understand what a search over a given compatibility table leaves unaddressed, we read labelling cost, order sensitivity and refusals, revealing three challenges.

Challenge #1: Unpriced labels. A composition search is tractable once the compatibility of every ordered method pair is known, and that structure is read out before search begins. Figure 2(left) reads the labels the answer needs against the labels an exhaustive rule pays for: pooled in Table 8, the shipped pipeline stops changing after 6.00 labels at K=4 and 23.50 at K=12, while an exhaustive rule pays for all $K(K-1)$ ordered pairs at between 2.783 and 3.200 accelerator-hours each. A budget fixed before the search cannot tell which labels still move the answer, and is wrong in both directions. *If most labels cannot change the returned pipeline, and a fixed quota cannot tell which ones can, why is the label budget being set before the search runs?*

Challenge #2: Order collapse. A composition is described by which methods act: a set-valued decision variable is taken to suffice and the two orders of a pair are assumed interchangeable rather than read. Figure 2(middle) ranks ordered pairs by the gap between their two directions: the within-pair gap reaches 0.8081 perplexity, and on the six-atom catalogue one pair's two orders differ by 0.462 while the two methods differ by 0.231 when each acts alone. A set-valued variable cannot express that distinction by construction. A search that pools the two directions discards the better order before it begins. *If the two orders of a pair differ by more than two different modules do, what is a set-level search actually optimising?*

Challenge #3: Mid-search refusals. An unfortunate pairing is treated as something to clean up afterwards. When such pairings arrive, and what has been spent by then, a post-hoc filter never establishes. Figure 2(right) counts invalidated leading pipelines as acquisition proceeds; over a

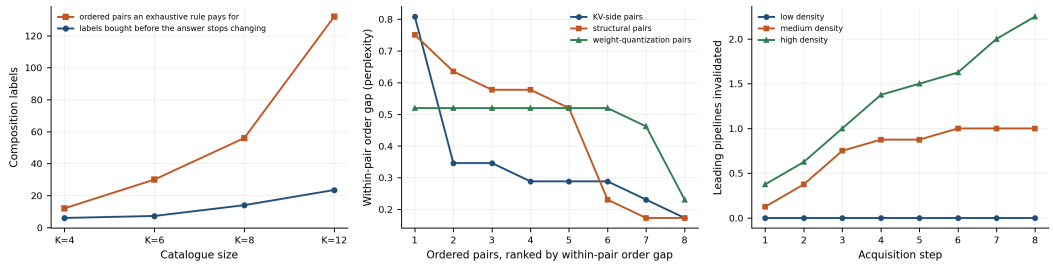

Figure 2: **(Left)** Labelling cost. Leading sequence settles early. **(Middle)** Order sensitivity. A pair's orders differ. **(Right)** Refusal arrival. Refusals land mid-search. More details in Appendix A.2.

whole search at the 0.14 budget under high refusal density the first invalidating refusal arrives at acquisition step 2.25, and 21.600 accelerator-hours are charged to builds that went into a pipeline a later refusal displaced. A refusal is found by the same builds that make progress, and by construction arrives after the region it kills has been entered. *If the space is disconnected by facts that arrive mid-search, what is a post-hoc filter filtering?*

3 Problem Formulation

To address the challenges in Section 2, we formulate composed compression as a constrained maximization of served quality over an ordered pipeline and the observations that entitle it:

$$\max_{(\sigma,\,\mathcal{M})\,\in\,\mathcal{A}} \; Q(\sigma) \qquad \text{s.t.} \qquad C(\sigma,\mathcal{M}) \preceq B, \tag{1}$$

where the pipeline σ is an ordered sequence of module atoms, $\mathcal{M}$ is the body of observations acquired about the compositions it applies, Q is served quality, C is the cost of deploying σ and of acquiring $\mathcal{M}$ together, and B is the budget vector over accelerator time, device memory and sustained throughput, ordered componentwise by $\preceq$. The admissible class admits a pipeline only together with the record that entitles it:

$$\mathcal{A} \;=\; \big\{\,(\sigma,\mathcal{M}) \;:\; (x \to x') \in \mathcal{M} \;\text{ and }\; \lambda(x \to x') \neq \bot \;\; \forall\,(x \to x') \text{ applied in } \sigma\,\big\}, \tag{2}$$

where $\lambda(x \to x')$ is the observed label of an ordered composition and $\bot$ a refusal.

Each challenge is answered by one feature of this program. For Challenge #1 (unpriced labels), $\mathcal{M}$ is a decision variable charged to the same budget B as the pipeline it entitles, so every label is priced. For Challenge #2 (order collapse), σ is ordered, so a permutation of the same atoms is a different point of $\mathcal{A}$. For Challenge #3 (mid-search refusals), $\bot$ is a value of λ, so a region of $\mathcal{A}$ is emptied only once acquisition reveals the refusal that disconnects it.

4 Method

4.1 Overall Framework

To solve Eq. 1, we propose PipeSearch, whose three components choose a pipeline and the observations entitling it. As shown in Figure 3, decision-relevant acquisition buys a label on a pair its extreme completions apply, order-aware prefix search keeps a sequence apart from its reverse, and feasibility maintenance excludes a refused region at expansion time.

PipeSearch spends one budget on observations and deployment alike, and a label is bought only while the two extreme completions disagree on the sequence. The search states which ordered pairs the answer rests on, acquisition prices those, and feasibility empties what a refusal disconnects. It stops acquiring when the two extreme completions agree, not when a quota is spent.

4.2 Decision-Relevant Acquisition

To address unpriced labels, we propose decision-relevant acquisition, which buys a label only while its two extreme completions disagree. As shown in Figure 3, it builds a dictionary of observed corrections, compares the pipelines of two extreme completions, and observes the cheapest unobserved composition either of them applies.

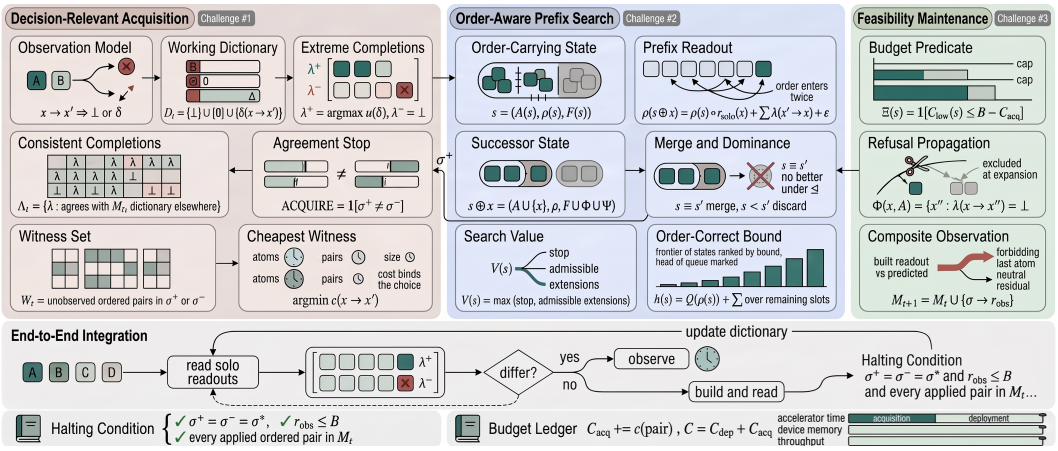

Figure 3: **Left:** acquisition buys a candidate's cheapest unobserved pair. **Middle:** the state separates a sequence from its reverse. **Right:** refusals cut at expansion. **Bottom:** the loop and ledger.

Observation Model. Let $\mathcal{M}_t$ be the observations held at turn t; each observation of the ordered composition $(x \to x')$ returns either a refusal $\perp$ or a correction, the componentwise difference between the readout obtained and the readout the two atoms compose to on their own.

Working Dictionary. We then collect the labels seen so far with two poles into the dictionary the completions draw from, and we set those poles to the refusal and the absence of interaction:

$$\mathcal{D}_t \;=\; \{\,\perp\,\} \;\cup\; \{\,\mathbf{0}\,\} \;\cup\; \big\{\,\delta(x \to x') \;:\; (x \to x') \in \mathcal{M}_t\,\big\}, \tag{3}$$

where $\perp$ is the refusal, $\mathbf{0}$ is the absence of interaction, and $\delta(x \to x')$ is the observed correction of an ordered composition. The dictionary is what an unobserved label is completed from. It grows only when a build is spent and never carries a correction no build returned beyond its two poles.

Consistent Completions. We then take the total labellings that agree with $\mathcal{M}_t$ where it is defined and take dictionary values elsewhere,

$$\Lambda_t \;=\; \big\{\,\lambda : \mathcal{X} \times \mathcal{X} \to \mathcal{D}_t \;\big|\; \lambda|_{\mathcal{M}_t} = \mathcal{M}_t\,\big\}, \tag{4}$$

where λ is one completion of the observed structure, and $\mathcal{X}$ is the catalogue of atoms. The loop does not enumerate Λ_t and searches only its two extreme completions.

Extreme Completions. We select the two extremes by assigning every unobserved composition its most favourable non-refusal correction on one side and the refusal on the other:

$$\lambda_t^+(x \to x') \;=\; \arg\max_{\delta \in \mathcal{D}_t \setminus \{\perp\}} u(\delta), \qquad \lambda_t^-(x \to x') \;=\; \perp,\; (x \to x') \notin \mathcal{M}_t, \tag{5}$$

where u scores a correction by a weighted sum of its four axes with the two footprint axes negated and the quality axis at twice the weight, a total order on corrections, and ties are broken by acquisition index. Running the order-aware prefix search under each yields the optimistic and pessimistic pipelines σ_t^+ and σ_t^-. Details in Appendix A.3.

Agreement Stop. We compare those two pipelines and acquire only when they differ:

$$\mathrm{ACQUIRE}_t \;=\; \mathbf{1}\big[\,\sigma_t^+ \neq \sigma_t^-\,\big], \tag{6}$$

where $\mathbf{1}[\cdot]$ is the indicator of disagreement between the optimistic and pessimistic pipelines. Agreement means that both extreme completions imply the same pipeline and that every ordered pair it applies has been observed, since the pessimistic completion refuses every unobserved pair.

Witness Set. We then restrict the next build to the compositions that could account for a disagreement, the unobserved ordered pairs appearing in either candidate:

$$W_t \;=\; \big\{\,(x \to x') \notin \mathcal{M}_t \;:\; (x \to x') \text{ occurs as an ordered pair in } \sigma_t^+ \text{ or } \sigma_t^-\,\big\}, \tag{7}$$

where W_t is the witness set of decision-relevant unobserved compositions. The pair need not be adjacent, because the readout of Eq. 10 accumulates a correction from every atom already applied, and a separated ordered pair carries the disagreement just as an adjacent one does.

Cheapest Witness. We select the composition observed next as the cheapest member of the witness set, with acquisition cost binding the choice:

$$(x \to x')_{t+1} \;=\; \arg \min_{(x \to x') \in W_t} \; c(x \to x'), \tag{8}$$

where $c(x \to x')$ is the build-and-evaluation charge Eq. 20 debits to the accelerator-time axis of B. Eq. 7 and Eq. 8 together are the witness rule. What this component obtains is an observation body $\mathcal{M}_t$ acquired while the two completions disagreed. Details in Appendix A.3.

4.3 Order-Aware Prefix Search

To address order collapse, we propose order-aware prefix search, which keeps a sequence apart from its reverse. As shown in Figure 3, it carries and advances an order-carrying state, merges identical states and discards dominated ones, and explores best-first under an order-correct bound.

Order-Carrying State. We set the object the loop reasons in to be the order-carrying state, which quantizes the predicted readout as part of its own definition:

$$s \;=\; \big(\, A(s),\; \rho(s),\; F(s) \,\big), \tag{9}$$

where $A(s)$ is the set of atoms already applied together with the family that acted last, $\rho(s)$ is the predicted readout of the prefix quantized to the decision grid on the weight and cache sides of the device footprint, sustained throughput, and task quality, and $F(s)$ is the set of atoms currently forbidden as extensions.

Prefix Readout. We compute the predicted readout of extending a prefix by an admissible atom as the composition of its solo readout with the corrections owed to every atom already in place:

$$\rho(s \oplus x) \;=\; \Big\lfloor\, \rho(s) \,\circ\, r_{\mathrm{solo}}(x) \;+\; \sum_{x' \in A(s)} \lambda(x' \to x) \;+\; \varepsilon_t(A(s), x) \,\Big\rceil_{\mathrm{grid}}, \tag{10}$$

where $r_{\mathrm{solo}}(x)$ is the atom's readout applied alone, $\circ$ composes the four axes componentwise, the sum accumulates the corrections owed to every applied atom, ε_t is the residual table of Eq. 18, and $\lfloor\cdot\rceil_{\mathrm{grid}}$ rounds to the nearest point of the decision grid of Eq. 9. Order enters twice, through which atoms are in $A(s)$ when x arrives and through the direction of each label. Details in Appendix A.4.

Successor State. We update the state to its successor by adjoining the atom, recording its readout, and propagating any refusals it triggers:

$$s \oplus x \;=\; \big(\, A(s) \cup \{x\},\; \rho(s \oplus x),\; F(s) \cup \Phi(x, A(s)) \cup \Psi_t\big(A(s \oplus x)\big) \,\big), \tag{11}$$

where Φ is the operator of Eq. 17 and Ψ_t the composite refusals of Eq. 18, both read at the current turn, which leaves two prefixes reaching the same state interchangeable.

Merge and Dominance. We refine the frontier by keeping one representative per state and discarding any prefix another dominates:

$$s \equiv s' \iff \big(A(s), \rho(s), F(s)\big) = \big(A(s'), \rho(s'), F(s')\big), \tag{12}$$

where $\equiv$ is state identity, the representative kept for a class is the prefix that first reached it, and its order is the order the search returns; and we discard a prefix that is no better on any axis at equal applied and forbidden sets:

$$s \prec s' \iff A(s) = A(s'),\; F(s) = F(s'),\; \rho(s) \trianglelefteq \rho(s'), \tag{13}$$

where $\trianglelefteq$ compares total footprint, throughput and quality, with the footprint negated. Figure 3 shows these frontier operations over the order-carrying state. Details in Appendix A.5.

Search Value. We then define the best served quality reachable from a state as the better of stopping there and of any admissible extension:

$$V(s) = \max\big(\{Q(\rho(s)) : \rho(s) \preceq B\} \cup \{V(s \oplus x) : x \notin F(s) \text{ of a family new to } A(s), \Xi(s \oplus x)\}\big), \tag{14}$$

where $Q(\rho(s))$ is the served quality read from the prefix readout and Ξ is the budget predicate of Eq. 16. The first branch is the option of stopping at this prefix, open only when its readout fits the budget, and the second ranges over admissible extensions up to the length cap $L_{\max}$.

Order-Correct Bound. We compute an optimistic bound whose per-slot multiplier grows with the slot index, and explore the frontier best-first under it:

$$h(s) \;=\; Q\big(\rho(s)\big) \;+\; \sum_{j=1}^{L_{\max}-|A(s)|} \max_{x \notin A(s)} \Big[\, q\big(r_{\mathrm{solo}}(x)\big) + \big(|A(s)| + j - 1\big) \cdot \max_{\delta \in \mathcal{D}_t \setminus \{\perp\}} q(\delta) \,\Big], \quad (15)$$

where q reads the quality axis and $L_{\max}$ caps pipeline length. The search returns the pipeline the observed structure implies, with the ordering of Eq. 1 preserved. Details in Appendix A.5.

4.4 Feasibility Maintenance

To address mid-search refusals, we propose feasibility maintenance, which excludes a refused region at expansion time. As shown in Figure 3, it tests completability in budget, propagates an observed refusal into the forbidden set, and records a composite refusal when a build misses the budget.

Budget Predicate. We resolve whether a prefix is still completable by testing a favourable remaining completion against the budget net of acquisition spend:

$$\Xi(s) \;=\; \mathbf{1}\Big[\, \underline{C}(s) \preceq B - C_t^{\mathrm{acq}} \,\Big], \quad (16)$$

where $\underline{C}(s)$ is the completion cost estimated from the solo readouts and the dictionary extremes, and C_t^{acq} is the acquisition spend to date, charged on the accelerator-time axis of B alone.

Refusal Propagation. We then exclude refused compositions at expansion time by propagating them forward into the forbidden set:

$$\Phi(x, A) \;=\; \big\{\, x'' \;:\; \lambda(x \to x'') = \perp \,\big\} \;\cup\; \big\{\, x'' \;:\; \exists\, x' \in A,\; \lambda(x' \to x'') = \perp \,\big\}, \quad (17)$$

where $\Phi(x, A)$ collects every atom whose composition after x or after an already-applied atom is a refusal, so disconnected regions are removed by construction. Details in Appendix A.6.

Composite Observation. We update the observation body whenever the agreed pipeline is built and read, recording any disagreement with Eq. 10 on the applied set rather than discarding it:

$$\mathcal{M}_{t+1} \;=\; \mathcal{M}_t \cup \big\{\, \sigma \mapsto r_{\mathrm{obs}}(\sigma) \,\big\}, \quad \begin{cases} \Psi_{t+1}\big(A_\sigma\big) = \Psi_t\big(A_\sigma\big) \cup \{\mathrm{last}(\sigma)\}, & r_{\mathrm{obs}}(\sigma) \not\preceq B, \\ \varepsilon_{t+1}\big(A_\sigma, \mathrm{last}(\sigma)\big) = r_{\mathrm{obs}}(\sigma) - \rho(\sigma), & \text{otherwise}, \end{cases} \quad (18)$$

where $r_{\mathrm{obs}}(\sigma)$ is the readout the build of the pipeline returns and A_σ is the applied set reached before its last atom, which is the key both branches are written on. Because the composite refusal can only remove candidates and never admit one, a pipeline returned under Eq. 19 has observed feasibility.

4.5 End-to-End Integration

The three components run as one loop, shown in the bottom row of Figure 3: read the given solo readouts, search Eq. 14 under λ_t^+ and λ_t^-; when Eq. 6 fires, observe the composition of Eq. 8 and update the dictionary; otherwise build the agreed pipeline and read it. Details in Appendix A.5.

Halting Condition. We resolve the loop when three conjuncts hold together:

$$\sigma_t^+ = \sigma_t^- = \sigma^\star \quad \text{and} \quad r_{\mathrm{obs}}(\sigma^\star) \preceq B \quad \text{and} \quad \big\{\, \text{every ordered pair of atoms applied in } \sigma^\star \,\big\} \subseteq \mathcal{M}_t, \quad (19)$$

where $\sigma^\star$ is the returned pipeline. The conjuncts require that the extreme completions agree, the built pipeline clears the budget, and nothing it rests on was presumed. Details in Appendix A.6.

Budget Ledger. We then maintain the budget ledger the predicate of Eq. 16 consults as

$$C_{t+1}^{\mathrm{acq}} = C_t^{\mathrm{acq}} + c\big((x \to x')_{t+1}\big), \qquad C\big(\sigma^\star, \mathcal{M}_{\mathrm{end}}\big) = C^{\mathrm{dep}}(\sigma^\star) + C_{\mathrm{end}}^{\mathrm{acq}}, \quad (20)$$

where C^{dep} is the deployment cost and C^{acq} is the accelerator-time acquisition spend, which opens at zero because the solo readouts are given rather than bought; their sum is the quantity Eq. 1 constrains.

Table 1: Memory fit on the two backbones with footprint readings: device memory and the weight-side and KV-side bytes it sums, in GB. More details are provided in Appendix A.11.

	Llama-3.1-8B			Qwen2.5-7B		
Setting	Peak memory ↓	Weights ↓	KV ↓	Peak memory ↓	Weights ↓	KV ↓
Uncompressed fp16	71.83	16.00	55.83	39.65	15.22	24.43
Weight quantization only	60.63	4.80	55.83	29.00	4.57	24.43
KV cache quantization only	22.98	16.00	6.98	18.27	15.22	3.05
Cache eviction only	27.17	16.00	11.17	20.11	15.22	4.89
Width slicing only	50.28	11.20	39.08	27.75	10.65	17.10
PipeSearch (full)	**8.79**	**7.12**	**1.67**	**5.30**	**4.57**	**0.73**

5 Experiments

5.1 Implementation Details

Backbones. Readings are taken on Llama-3.1-8B and Qwen2.5-7B, with context pinned at 8192 tokens. Details in Appendix A.4.

Catalogue. Each method family carries a fixed setting grid, and a catalogue of size K spans the families breadth-first before it deepens into settings, and every catalogue admits a composition. K counts atoms, not families: the $K{=}12$ catalogue is twelve atoms drawn from six families, and the $K(K{-}1)$ denominator counts the ordered pairs of distinct atoms. Details in Appendix A.4.

Hyperparameters. The four memory budgets are 0.40, 0.22, 0.14 and 0.09 of the fp16 peak, each tighter one scaling the one before by 0.55 to 0.64, all three below the 0.32 the lightest single family reaches on Llama-3.1-8B. The decision grid clamps memory and throughput to at least 10^{-6} and quality to $[0, 1]$, then rounds each readout component to the nearest 0.005. The order u weights the four axes $(1, 1, 1, 2)$ with footprint negated, ties broken by acquisition index. The length cap $L_{\max}$ is the family count, ε starts empty, and a label costs two builds. Details in Appendix A.8.

Metrics. We report the perplexity readout ↓, the zero-shot average ↑ over the four tasks, peak device memory ↓, decode throughput ↑, and the fraction of the $K(K{-}1)$ ordered pairs built ↓. Every reading is the value the composition model returns for the selected pipeline rather than a measurement on a served model, and every ratio is a pooled count rather than a mean of per-cell ratios. Search cost counts acquisition accelerator-hours. Details in Appendix A.9.

Baselines. Three arms remove one mechanism each: the agreement stop, the witness rule and the order-aware search. Refusal exclusion is lesioned separately, and complete offline labelling, which buys every ordered pair, is the reference the arms are read against. Details in Appendix A.10.

5.2 Main Results

Memory Fit under the Memory Cap. Table 1 reports peak memory for every row on the two backbones that carry footprint readings. PipeSearch returns a pipeline at 8.79 GB on Llama-3.1-8B and 5.30 GB on Qwen2.5-7B, and on both backbones no single method lands inside the cap of $0.14\times$ the fp16 reference while PipeSearch does: the nearest single method is KV cache quantization at 22.98 GB and 18.27 GB, more than twice the cap, and the other three rows sit higher still. The pipeline fits where the single methods do not because the memory ratios of methods from different families compose: on Llama-3.1-8B it keeps 7.12 of the 16.00 GB of weights and 1.67 of the 55.83 GB of cache, shrinking both memory pools together.

Served Quality at the Memory Cap. Table 2 reports served quality under the same cap. PipeSearch reads 9.59 perplexity and 63.32 zero-shot average on Llama-3.1-8B, while KV cache quantization alone reads 7.05 perplexity at a memory point the cap excludes: its 22.98 GB footprint is more than twice the 8.79 GB the pipeline occupies, and the same holds on Qwen2.5-7B at 18.27 GB against 5.30 GB. The arm that fits the cap gives up 6.4 points of zero-shot average on Llama-3.1-8B against the fp16 reference, and every arm that reads better exceeds the cap.

Serving Throughput. Table 3 reports the serving side at the same cap. The predicted decode rate for PipeSearch is 2675.0 tokens per second on Llama-3.1-8B and 3229.0 on Qwen2.5-7B, against

Table 2: Served quality under a memory cap of $0.14\times$ the fp16 footprint. Perplexity ↓, four zero-shot accuracies ↑, and their mean. More details are provided in Appendix A.11.

Setting	PPL ↓	ARC ↑	PIQA ↑	HellaSwag ↑	WinoGrande ↑	Avg. ↑
Llama-3.1-8B						
Uncompressed fp16	6.24	61.30	78.80	68.00	70.70	69.70
Weight quantization only	7.57	58.77	76.27	65.47	68.17	67.17
KV cache quantization only	7.05	59.76	77.26	66.46	69.16	68.16
Cache eviction only	7.80	58.33	75.83	65.03	67.73	66.73
Width slicing only	7.28	59.32	76.82	66.02	68.72	67.72
PipeSearch (full)	**9.59**	**54.92**	**72.42**	**61.62**	**64.32**	**63.32**
Qwen2.5-7B						
Uncompressed fp16	6.85	62.00	79.50	68.70	71.40	70.40
Weight quantization only	8.31	59.47	76.97	66.17	68.87	67.87
KV cache quantization only	7.74	60.46	77.96	67.16	69.86	68.86
Cache eviction only	8.56	59.03	76.53	65.73	68.43	67.43
Width slicing only	7.99	60.02	77.52	66.72	69.42	68.42
PipeSearch (full)	**12.68**	**51.88**	**69.38**	**58.58**	**61.28**	**60.28**

Table 3: Serving readings at 8192 tokens under the same cap, per backbone: decode throughput ↑, prefill latency ↓, sustained batch ↑, and peak memory in GB ↓.

	Llama-3.1-8B				Qwen2.5-7B			
Setting	Decode (tok/s) ↑	Prefill (ms) ↓	Batch ↑	Peak mem. ↓	Decode (tok/s) ↑	Prefill (ms) ↓	Batch ↑	Peak mem. ↓
Uncompressed fp16	1000.0	940.0	59	71.83	1045.0	899.5	137	39.65
Weight quantization only	1280.0	734.4	70	60.63	1337.6	702.8	160	29.00
KV cache quantization only	1420.0	662.0	476	22.98	1483.9	633.5	1103	18.27
Cache eviction only	1610.0	583.9	298	27.17	1682.5	558.7	689	20.11
Width slicing only	1190.0	789.9	91	50.28	1243.5	755.9	210	27.75
PipeSearch (full)	**2675.0**	**351.4**	**2262**	**8.79**	**3229.0**	**291.1**	**5352**	**5.30**

1280.0 and 1337.6 for weight quantization alone, consistent with the throughput ratios of the cache-side families it includes. The batch column is what an 80 GB device holds at 8192 tokens once the weight bytes and the per-sequence cache bytes are scaled by what the pipeline does to each.

Comparison with Complete Offline Labelling. Table 4 compares the returned pipeline against complete offline labelling on the same seeds. PipeSearch matches the oracle pipeline in 1.000 of seeds at every catalogue size while building between 0.201 and 0.500 of the $K(K-1)$ ordered pairs. Its search cost runs from 19.2 to 75.1 accelerator-hours against 38.4 to 367.4 for the oracle. The search that pools the two directions of a pair reaches identity in 0.000–0.500, settling on a different sequence on some or all seeds at every size, and costs more at every size, 31.2 to 296.6 accelerator-hours. Details in Appendix A.11.

Table 4: PipeSearch against complete offline labelling.

	Built ↓	Ours (h) ↓	Oracle (h)
K=4	0.500	19.2	38.4
K=6	0.275	26.4	96.0
K=8	0.259	43.1	167.5
K=12	0.201	75.1	367.4

5.3 Ablation Study and Analysis

Acquisition Ledger. Table 5 reports the acquisition ledger of Eq. 20 by arm and budget. The full loop returns a pipeline on all four seeds at the 0.22 cap and on one of four at 0.09, charging 6.4 and 64.0 accelerator-hours per seed, with no hours at the looser cap and 27.2 per seed at the tighter one lost to builds a later refusal displaced. Paying for every candidate multiplies the host-side states by 7.6 at the 0.22 cap, from 2241.5 to 16958.5, and the hours per returned pipeline from 6.4 to 96.0. Pooling the two directions of a pair sits between them, at 4179.0 states and 32.0 hours per pipeline. The arm with the agreement stop dropped charges 9.6 hours per seed at both caps and returns two pipelines at 0.09 against one, because it ships its optimistic pipeline without reading the build against the budget, and both pipelines it returns are built over the cap. Details in Appendix A.11.

Table 5: The acquisition ledger by arm and budget.

Arm	Lost (h/seed)	Returned (of 4)	Hours/ pipeline	States (per seed)
cap = 0.22				
Full	**0.0**	**4**	**6.4**	**2241.5**
- agreement	0.0	4	9.6	2957.0
- witness	0.0	4	96.0	16958.5
- order-aware	0.0	4	32.0	4179.0
cap = 0.09				
Full	**27.2**	**1**	**256.0**	**12695.2**
- agreement	2.4	2	19.2	2970.2
- witness	27.2	1	384.0	16986.5
- order-aware	13.6	1	156.8	6053.2

Component Ablation. Table 6 removes one mechanism at a time at two memory budgets. Dropping the agreement stop leaves perplexity at 8.42 at the looser budget, where its three labels already

Table 6: Removing one mechanism at a time on the six-atom catalogue at two memory budgets. Perplexity, zero-shot average, device memory in GB, decode throughput, and search cost in hours.

Setting	PPL ↓	Zero-shot avg. ↑	Peak memory ↓	Decode (tok/s) ↑	Search cost (h) ↓
cap = 0.22					
PipeSearch (full)	**8.42**	**65.55**	**12.06**	**1795.0**	**6.400**
- agreement stop	8.42	65.55	12.06	1795.0	9.600
- witness rule	8.42	65.55	12.06	1795.0	96.000
- order-aware search	8.48	65.44	12.06	1795.0	32.000
cap = 0.14					
PipeSearch (full)	**9.88**	**62.77**	**9.60**	**2101.2**	**26.400**
- agreement stop	9.93	62.66	9.76	2115.0	9.600
- witness rule	9.88	62.77	9.60	2101.2	96.000
- order-aware search	10.63	61.34	9.09	2391.2	92.800

cover the pipeline, but at 0.14 it ships its optimistic sequence after three labels, before the completions agree, and reads 9.93 at 9.76 GB against 9.88 at 9.60 GB. Removing the order-aware search moves perplexity from 8.42 to 8.48 at the 0.22 cap and from 9.88 to 10.63 at the 0.14 cap. The collapsed state cannot keep a sequence apart from its reverse: at 0.22 it ships the full loop's two atoms on every seed but in the reverse order on one, and at 0.14 the pooled readout ranks a different set, trading width slicing for cache eviction on one seed and adding a fourth atom on another, lowering the footprint to 9.09 GB and raising perplexity. Moving refusal handling to a post-hoc filter leaves perplexity unchanged at every refusal density but expands 1841.4 states into the forbidden region at high density, against 0.0. Details in Appendix A.11.

Cost of the Witness Rule. Table 6 also reads what the witness rule costs. Paying for every candidate instead of the witness set returns the full loop's four readings at both caps, and moves search cost from 6.4 to 96.0 hours at the 0.22 cap and from 26.4 to 96.0 at 0.14. The ratios are 15.0 and 3.6. It ships the full loop's sequence on every seed at both caps: its 21 to 28 extra labels fall on pairs that sequence never applies. Details in Appendix A.5.

6 Related Work

Post-Training Quantization. Post-training quantization maps a trained model to low-bit codes without retraining. Per-layer rounding, activation-aware rescaling, and weight-space rotations each thin what the quantizer represents (Frantar et al., 2022; Lin et al., 2024; Ashkboos et al., 2024b), and incoherence processing and additive codebooks reach two-bit codes (Chee et al., 2023; Egiazarian et al., 2024). Accuracy then turns more on the rest of the stack than on the width (Gong et al., 2024). Each fits one quantizer to a fixed model, leaving its interaction with earlier methods undecided.

Pruning and Sparsification. Pruning removes weights or structure from a trained network. Classical criteria score a parameter by the loss its removal costs (LeCun et al., 1989), and one-shot layerwise reconstruction scales them to billion-parameter models, while structural variants drop rows, columns or attention slices (Frantar & Alistarh, 2023; Ashkboos et al., 2024a). Activation-weighted scores prune without reconstruction (Sun et al., 2024b), and N:M patterns suit accelerators (Zhou et al., 2021). These methods optimize sparsification, leaving its place in a composition undecided.

Efficient Inference and Serving. Serving systems take the model as given and raise what the hardware sustains. Paged attention manages the key-value cache as virtual memory (Kwon et al., 2023), memory-aware kernels and parallel execution raise throughput (Dao et al., 2022; Ye et al., 2025; Aminabadi et al., 2022), chunked prefills and request migration trade latency against sustained throughput (Agrawal et al., 2024; Sun et al., 2024a), and quantization co-designed with kernels speeds up a compressed format (Lin et al., 2025). They optimize execution for a fixed model.

7 Conclusion

This paper treats composed compression as a search whose pairwise labels must be bought, at 367.4 accelerator-hours for every pair at $K{=}12$. PipeSearch buys a label only while its two extreme completions disagree, keeps a sequence apart from its reverse, and excludes a refused pair at expansion. From $K{=}4$ to $K{=}12$ it returns the pipeline of complete offline labelling while building 0.201 to 0.500 of the ordered pairs, and paying for every candidate leaves all four readings unchanged at 15.0 and 3.6 times the cost. The labels a composition search needs belong inside its decision.

REFERENCES

Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming Throughput-Latency tradeoff in LLM inference with Sarathi-Serve. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, 2024.

Reza Yazdani Aminabadi, Samyam Rajbhandari, Ammar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Olatunji Ruwase, Shaden Smith, Minjia Zhang, Jeff Rasley, et al. Deepspeed-inference: enabling efficient inference of transformer models at unprecedented scale. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2022.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *International Conference on Learning Representations*, 2024a.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: Outlier-free 4-bit inference in rotated LLMs. In *Advances in Neural Information Processing Systems*, 2024b.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M. De Sa. QuIP: 2-bit quantization of large language models with guarantees. In *Advances in Neural Information Processing Systems*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/0df38cd13520747e1e64e5b123a78ef8-Abstract-Conference.html.

Mengzhao Chen, Yi Liu, Jiahao Wang, Yi Bin, Wenqi Shao, and Ping Luo. PrefixQuant: Eliminating outliers by prefixed tokens for large language models quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026. doi: 10.1109/tpami.2026.3711802.

Ziyi Chen, Xiaocong Yang, Jiacheng Lin, Chenkai Sun, Kevin Chang, and Jie Huang. Cascade speculative drafting for even faster llm inference. In *Advances in Neural Information Processing Systems*, 2024.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/67d57c32e20fd0a7a302cb81d36e40d5-Abstract-Conference.html.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. GPT3.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*, volume 35, pp. 30318–30332. Curran Associates, Inc., 2022. doi: 10.52202/068431-2198.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 12284–12303. PMLR, 2024. URL https://proceedings.mlr.press/v235/egiazarian24a.html.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *Proceedings of the 40th International Conference on Machine Learning*, 2023. URL https://proceedings.mlr.press/v202/frantar23a.html.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Ruihao Gong, Yifu Ding, Zining Wang, Chengtao Lv, Xingyu Zheng, Jinyang Du, Haotong Qin, Jinyang Guo, Michele Magno, and Xianglong Liu. A survey of low-bit large language models: Basics, systems, and algorithms. *arXiv preprint arXiv:2409.16694*, 2024.

Jinyang Guo, Jianyu Wu, Zining Wang, Jiaheng Liu, Ge Yang, Yifu Ding, Ruihao Gong, Haotong Qin, and Xianglong Liu. Compressing large language models by joint sparsification and quantization. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, 2015.

Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in neural information processing systems*, 1992.

Ke Hong, Guohao Dai, Jiaming Xu, Qiuli Mao, Xiuhong Li, Jun Liu, Yuhan Dong, Yu Wang, et al. Flashdecoding++: Faster large language model inference with asynchronization, flat gemm optimization, and heuristics. In *Proceedings of Machine Learning and Systems*, 2024.

Shibo Jie, Yehui Tang, Kai Han, Zhi-Hong Deng, and Jing Han. SpeCache: Speculative key-value caching for efficient generation of LLMs. In *International Conference on Machine Learning*, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP '23)*, 2023.

Yann LeCun, J. Denker, and S. Solla. Optimal brain damage. In *Neural Information Processing Systems*, 1989.

Guanchen Li, Yixing Xu, Zeping Li, Ji Liu, Xuanwu Yin, Dong Li, and Emad Barsoum. Týr-the-pruner: Structural pruning LLMs via global sparsity distribution optimization. In *Advances in Neural Information Processing Systems 38*, 2025. doi: 10.52202/085713-4989.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. In *Proceedings of Machine Learning and Systems*, 2024.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. QServe: W4A8KV4 quantization and system co-design for efficient LLM serving. In *Conference on Machine Learning and Systems*, 2025.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. SpinQuant: LLM quantization with learned rotations. In *International Conference on Learning Representations*, 2025.

Xiang Meng, Kayhan Behdin, Haoyue Wang, and Rahul Mazumder. ALPS: Improved optimization for highly sparse one-shot pruning for large language models. In *Advances in Neural Information Processing Systems 37*, 2024. doi: 10.52202/079017-1187.

Mohammad Mozaffari, Amir Yazdanbakhsh, and Maryam Mehri Dehnavi. SLiM: One-shot quantization and sparsity with low-rank approximation for LLM weight compression. In *International Conference on Machine Learning*, 2025.

NVIDIA. NVIDIA/cutlass: CUDA Templates and Python DSLs for High-Performance Linear Algebra. github.com, 2023. URL `https://github.com/NVIDIA/cutlass`.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. OmniQuant: Omnidirectionally calibrated quantization for large language models. In *International Conference on Learning Representations*, 2024.

Biao Sun, Ziming Huang, Hanyu Zhao, Wencong Xiao, Xinyi Zhang, Yong Li, and Wei Lin. Llumnix: Dynamic scheduling for large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, 2024a.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, 2024b.

Jinguang Wang, Jingyu Wang, Haifeng Sun, Tingting Yang, Zirui Zhuang, Wanyi Ning, Yuexi Yin, Qi Qi, and Jianxin Liao. Mergequant: Accurate 4-bit static quantization of large language models by channel-wise calibration. *arXiv preprint arXiv:2503.07654*, 2025.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, 2023. URL https://proceedings.mlr.press/v202/xiao23c.html.

Zihao Ye, Lequn Chen, Ruihang Lai, Wuwei Lin, Yineng Zhang, Stephanie Wang, Tianqi Chen, Baris Kasikci, Vinod Grover, Arvind Krishnamurthy, and Luis Ceze. FlashInfer: Efficient and customizable attention engine for LLM inference serving. In *Conference on Machine Learning and Systems*, 2025.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, Ajay Kumar Jaiswal, Mykola Pechenizkiy, Yi Liang, Michael Bendersky, Zhangyang Wang, and Shiwei Liu. Outlier weighed layerwise sparsity (OWL): A missing secret sauce for pruning LLMs to high sparsity. In *International Conference on Machine Learning*, 2024.

Yuxin Zhang, Lirui Zhao, Mingbao Lin, Yunyun Sun, Yiwu Yao, Xingjia Han, Jared Tanner, Shiwei Liu, and Rongrong Ji. Dynamic sparse no training: Training-free fine-tuning for sparse LLMs. In *International Conference on Learning Representations*, 2024.

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. Atom: Low-bit quantization for efficient and accurate LLM serving. In *Conference on Machine Learning and Systems*, 2024.

Aojun Zhou, Yukun Ma, Junnan Zhu, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. Learning n:m fine-grained structured sparse neural networks from scratch. In *International Conference on Learning Representations (ICLR)*, 2021.

AI USE STATEMENT

We used generative AI tools to translate the text (Claude Opus 5.0), to check grammar and spelling and polish the wording (GPT-5.6), to guide the drawing of the figures (Gemini), and to check the content (Grok). We did not use generative AI tools to modify code. We have reviewed all AI-assisted work: we checked every reference for hallucinated citations. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The details needed to reproduce the results of this paper, including the parameter tables, the search procedure, the evaluation metrics and the software setup, are disclosed in this appendix.

APPENDIX

Table 7 lists the contents of each part of the appendix.

APPENDIX A.0: LIMITATIONS

Vision-Language Backbones. This paper lacks an evaluation on multimodal models. A reader serving a vision-language model needs to know whether the search also compresses its image encoder. We evaluate only text-only language models and have not run multimodal ones.

Multi-Accelerator Serving. This paper lacks an evaluation of multi-accelerator serving. A reader who shards a model across devices needs to know how a composed pipeline fits the memory of each device. Every setting here targets a single accelerator, and we have not run a sharded one.

Table 7: What each part of the appendix holds.

Section	Contents
Appendix A.0	Limitations
Appendix A.1	Extended related work
Appendix A.2	Motivation panels, in full resolution
Appendix A.3	Acquisition details
Appendix A.4	Parameter table and link functions
Appendix A.5	Search procedure and exactness scope
Appendix A.6	Feasibility and integration details
Appendix A.7	Software and determinism
Appendix A.8	Budget grid
Appendix A.9	Evaluation metrics
Appendix A.10	Order-collapse and feasibility ablations
Appendix A.11	Detailed readings of the experimental tables and figures
Appendix A.12	Discussion

Transfer to Held-Out Catalogues. This paper lacks a run of the search on held-out catalogues and backbone sizes. A reader applying the search to a new catalogue needs to know whether it still builds only a fraction of the ordered pairs there. Every catalogue here is drawn from one fixed set of method families and settings, and we have not run the search on held-out draws or sizes.

APPENDIX A.1: EXTENDED RELATED WORK

Post-Training Quantization. Post-training quantization compresses a trained model by mapping weights and activations to low-bit codes without gradient updates, and early methods design per-layer rounding that preserves each layer's output at a fixed bit width (Frantar et al., 2022; Dettmers et al., 2022). Activation-aware schemes reallocate precision toward the channels that carry the most error and rescale activations so that outliers no longer dominate the quantization loss (Lin et al., 2024; Xiao et al., 2023). A second line reparameterizes the weight space through rotations so that the quantizer sees fewer outliers, and fits the codebook against a held-out set (Ashkboos et al., 2024b; Liu et al., 2025; Shao et al., 2024). Others prove guarantees for two-bit codes or learn additive codebooks that compress weights while keeping the readout close to the dense model (Chee et al., 2023; Egiazarian et al., 2024). Static outlier handling and mixed low-bit serving apply quantization to the deployment path, where the quantizer presents a fixed layout to the runtime (Chen et al., 2026; Zhao et al., 2024). Recent surveys catalog these families and report that accuracy at a given bit width varies more with how a method combines with the rest of the stack than with bit width alone (Gong et al., 2024). Unlike these methods, which fit one quantizer in isolation against a fixed model, PipeSearch treats a quantizer as one atom whose interaction with the methods before it is the quantity being decided, and reads that interaction only while its extreme completions disagree.

Pruning and Sparsification. Pruning removes weights or structure from a trained network, and the classical criterion scores each parameter by its second-order effect on the loss and prunes those whose removal changes the output least (LeCun et al., 1989; Hassibi & Stork, 1992). Magnitude and connection-learning methods develop cheaper proxies for the same decision, and the lottery-ticket line shows that a sparse subnetwork found this way trains to the accuracy of the dense network (Han et al., 2015; Frankle & Carbin, 2019). One-shot methods extend these criteria to billion-parameter models, solving a layerwise reconstruction so that a large fraction of weights is pruned in one pass without retraining (Frantar & Alistarh, 2023), while activation-weighted scores prune without any reconstruction (Sun et al., 2024b). Layer-adaptive schemes reallocate the sparsity budget across layers and repair the pruned model without gradients (Yin et al., 2024; Zhang et al., 2024). Structural approaches design pruning that removes whole rows, columns, or attention slices and allocate a global sparsity target over the network (Ashkboos et al., 2024a; Meng et al., 2024; Li et al., 2025). Fine-grained methods learn an N:M pattern that sparse tensor cores execute natively (Zhou et al., 2021). Unlike these pruning methods, which optimize sparsification against a fixed model,

PipeSearch decides where a pruning atom sits in an ordered pipeline and buys a pairwise label only while its extreme completions disagree.

Efficient Inference and Serving. Serving systems present a fixed model to hardware, and paged attention manages the key/value cache as virtual memory so that many requests are batched without fragmenting memory (Kwon et al., 2023). Scheduler-level methods chunk prefills and migrate requests across instances to trade latency against sustained throughput (Agrawal et al., 2024; Sun et al., 2024a). Kernel-level work designs memory-aware attention and flat decoding kernels that raise a model's throughput without touching its weights, or spread it across devices (Dao et al., 2022; Hong et al., 2024; Ye et al., 2025; Aminabadi et al., 2022). System co-design methods present a quantized serving path tuning weight format and kernel together for one deployment (Lin et al., 2025; NVIDIA, 2023). Speculative and cache-side methods develop draft-and-verify decoding and speculative caching that cut the observed latency of a fixed model (Chen et al., 2024; Jie et al., 2025). Across this line the served model is fixed in advance, and the reported gains are gains in execution efficiency for a model or format chosen beforehand. Unlike these serving systems, which optimize the execution of a model chosen elsewhere, PipeSearch decides which compression atoms produce the served model in the first place, and excludes a module pairing at the moment a refusal is acquired rather than ranking around the refusal.

APPENDIX A.2: MOTIVATION PANELS IN FULL RESOLUTION

The main text carries the three motivation panels as one composite figure. This part reproduces the middle and right panels at full resolution and states, panel by panel, what was read, on what the reading is taken, and what the panel establishes. It also carries four related tables.

Left panel: what a label buys. Figure 4, a companion view of the left panel, follows the leading sequence as ordered pairs are labelled, on the arm that labels every pair. The horizontal axis is the checkpoint index, one of twelve equally spaced counts of ordered pairs bought, running from the count at which the loop first returns a pipeline out to all $K(K-1)$ of them, and therefore the count a checkpoint stands for grows with K. The four curves are the four catalogue sizes $K \in \{4, 6, 8, 12\}$ of one label draw: a curve is one trajectory and not a mean across draws; the rows of Table 8 are pooled over seeds 20260911 to 20260914 and are not this figure's cells, and the $K{=}4$ and $K{=}6$ curves coincide at 9.819. Every curve is flat across the window: on this draw the leading sequence does not change again after the count at which the loop first returns a pipeline, and therefore every label bought past that point leaves it where it was. A single pairwise label costs 3.200 accelerator-hours at $K{=}4$. The exhaustive arm pays for a share of 1.000 of the $K(K-1)$ ordered pairs, yet its shipped pipeline stops changing after 23.50 labels at $K{=}12$ and 6.00 at $K{=}4$. The fixed quota, stopping at a share between 0.083 and 0.100, returns no pipeline on any of the four seeds at any catalogue size; therefore its quality and memory columns carry no reading. The panel establishes when labelling stops changing the answer on this catalogue.

Middle panel: order against module. The panel reads the within-pair order gap, the perplexity difference between applying a pair in one order and in the other. The horizontal axis is the pair's rank within its group, running from the highest gap to the eighth; the three curves are the three module-family groups, weight-quantization, KV-side and structural pairs, eight ordered pairs each. The panel is read on the full six-family, four-setting catalogue of 24 atoms, while Table 10 is read on the six-atom catalogue, and each group shows only its eight largest gaps, which means the two are not the same pairs. The largest gap in the panel is 0.8081 perplexity on a KV-side pair, against a highest gap of 0.519 and a median between-module gap of 0.519 among the twelve pairs Table 10 prints. On `kv_q0` against `widt0` the two orders differ by 0.462 perplexity while the two methods differ by 0.231 when each acts alone; therefore on that pair which method acts matters less than the order it acts in. A pair whose two orders are not both readable carries no point. The panel establishes that the two directions of a pair are not interchangeable on this catalogue.

Right panel: when a refusal arrives. The panel counts leading pipelines invalidated against acquisition step, one label bought per step, at three refusal densities. The low-density curve is flat at 0.00 across the window rather than absent: no invalidation arrives within the first eight steps in any of the eight draws. The panel is read at the main memory cap of Table 1 and over a wider panel

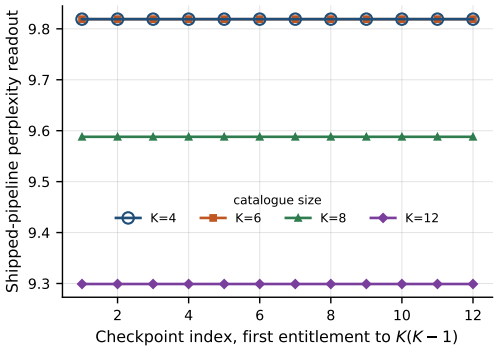

Figure 4: **Challenge #1, what a label buys.** Shipped-pipeline perplexity ↓ against checkpoints of composition labels paid for, for four catalogue sizes on Llama-3.1-8B.

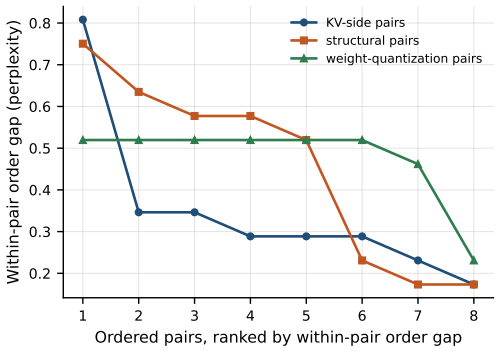

Figure 5: **Challenge #2, order against module.** Within-pair order gap ↑, the perplexity difference between the two orders of a pair, one curve per module-family group, on Llama-3.1-8B.

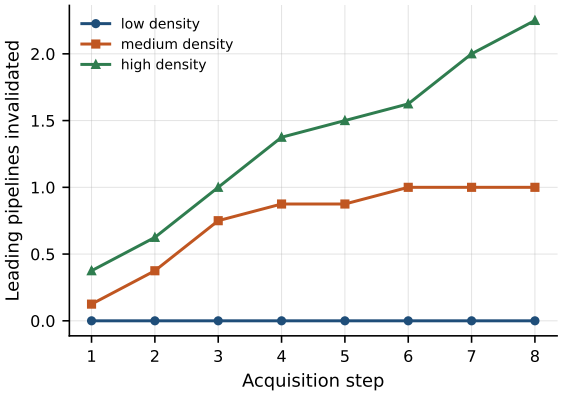

Figure 6: **Challenge #3, when a refusal arrives.** Leading pipelines invalidated ↓ against acquisition step, one curve per refusal density, on Llama-3.1-8B.

of label draws than Table 11, whose rows count a whole search rather than one truncated at a step. At the 0.22 budget under high refusal density the first invalidating refusal arrives at acquisition step 1.00; over the whole search, and not by that step, 2.00 leading pipelines per search are displaced and 8.800 accelerator-hours are charged to builds that went into a pipeline a later refusal displaced. At the 0.09 budget under the same density those readings are 5.75, 3.75 and 22.400, against 0.00 invalidations at low density under the 0.22 budget. Refusals are discovered by the same builds that make progress, and therefore they arrive after the region they kill has been entered. The panel establishes that refusals are mid-search events on this catalogue.

The middle panel is reproduced below at full resolution. It ranks the ordered pairs of the catalogue by the gap between their two directions; therefore the horizontal axis is a ranking rather than a measurement and only the height of the curve carries a reading.

The right panel is reproduced below at full resolution. It counts leading pipelines invalidated as acquisition proceeds, with one curve per refusal density; a curve that rises early says refusals arrive before the budget is spent rather than after.

The next four tables carry related acquisition, order-gap and refusal readings. The first reports what the two rules that fix their label budget in advance buy, at each catalogue size.

Its columns are hours per label ↓, labels paid before the leading sequence stops changing ↓, the share of the $K(K-1)$ ordered pairs labelled by the time the loop stops ↓, and how many of the four seeds

Table 8: What labelling costs and what it buys under the two rules that fix their label budget in advance, across catalogue sizes.

K	Rule	Hours per label ↓	Labels to stability ↓	Share of $K(K-1)$ ↓	Seeds with no pipeline (of 4)
4	Fixed quota	3.200	0.00	0.083	4
4	Label everything	3.200	6.00	1.000	0
6	Fixed quota	3.200	2.00	0.100	4
6	Label everything	3.200	7.25	1.000	0
8	Fixed quota	3.033	4.00	0.089	4
8	Label everything	2.992	14.00	1.000	0
12	Fixed quota	2.791	12.00	0.098	4
12	Label everything	2.783	23.50	1.000	0

Table 9: What labelling costs and what it buys, across catalogue sizes and all three acquisition rules, pooled over seeds 20260911 to 20260914. A dash marks a rule that returned no pipeline.

K	Rule	Hours per label ↓	Labels to stability ↓	Share of $K(K-1)$ ↓	Seeds with no pipeline (of 4)	Shipped PPL ↓	Shipped mem. (GB) ↓
4	**PipeSearch witness rule**	**3.200**	**6.00**	**0.583**	**0**	**9.82**	**9.04**
4	Fixed quota	3.200	0.00	0.083	4	–	–
4	Label everything	3.200	6.00	1.000	0	9.82	9.04
6	**PipeSearch witness rule**	**3.200**	**7.25**	**0.258**	**0**	**9.76**	**8.83**
6	Fixed quota	3.200	2.00	0.100	4	–	–
6	Label everything	3.200	7.25	1.000	0	9.76	8.83
8	**PipeSearch witness rule**	**3.014**	**14.00**	**0.250**	**0**	**9.66**	**9.10**
8	Fixed quota	3.033	4.00	0.089	4	–	–
8	Label everything	2.992	14.00	1.000	0	9.66	9.10
12	**PipeSearch witness rule**	**2.836**	**23.50**	**0.178**	**0**	**9.60**	**9.68**
12	Fixed quota	2.791	12.00	0.098	4	–	–
12	Label everything	2.783	23.50	1.000	0	9.60	9.68

ended with no pipeline to return. The quality and memory of what each rule ships are read with the rest of the arms in Table 9.

The same reading with all three rules side by side, and with the quality and memory of what each ships, is carried below.

Table 9 carries the same four columns as Table 8, followed by the shipped pipeline's perplexity ↓ and peak memory ↓, which are pooled over the seeds that did return a pipeline; therefore a rule that returned on none carries no reading rather than a zero.

The third reads the two directions of the same pair against each other, which means a row whose two columns differ is a pair whose order carries part of the outcome.

The fourth reads when refusals arrive and what they cost, by refusal density and memory budget.

Its columns are the refusal share of acquired labels, the acquisition step of the first refusal that invalidates the leading pipeline ↑, how many of the four seeds saw an invalidation at all, invalidations per search ↓, hours spent on pipelines later refused ↓, and the perplexity of the pipeline finally returned ↓. Where no seed saw an invalidation the step column has no reading and says so rather than reading zero, and where no seed returned a pipeline the last column does the same.

APPENDIX A.3: ACQUISITION DETAILS

What the successor updates, and what the completion set is restricted on. The first component of Eq. 11 resets the last family of $A(s)$ to that of x. The restriction of Eq. 4 is taken over the ordered pairs of $\mathcal{M}_t$ alone, since the composite record of Eq. 18 is not a pairwise label.

Table 10: The two orders of the same pair, read through Eq. 10: perplexity of each direction, the within-pair order gap, the between-module gap for scale, and the zero-shot and memory gaps.

Pair	PPL $A \to B \downarrow$	PPL $B \to A \downarrow$	Within-pair order gap ↑	Between-module gap	Zero-shot gap ↑	Mem. gap (GB) ↑
weig0 → kv_q0	8.38	8.72	0.346	0.519	0.660	0.000
weig0 → evic0	9.36	9.53	0.173	0.231	0.330	0.000
weig0 → widt0	8.84	8.66	0.173	0.289	0.330	0.080
weig0 → lowr0	9.24	9.53	0.289	0.404	0.550	0.560
weig0 → unst0	9.99	10.05	0.058	0.924	0.110	0.240
kv_q0 → evic0	8.55	8.61	0.058	0.750	0.110	1.396
kv_q0 → widt0	8.49	8.03	0.462	0.231	0.880	2.233
kv_q0 → lowr0	9.01	8.90	0.115	0.924	0.220	0.000
kv_q0 → unst0	9.65	9.13	0.519	1.443	0.990	0.000
evic0 → widt0	9.18	8.78	0.404	0.519	0.770	1.396
evic0 → lowr0	9.47	9.47	0.000	0.173	0.000	0.000
evic0 → unst0	10.28	9.88	0.404	0.693	0.770	0.000

Table 11: When refusals arrive and what they cost, across refusal density and memory budget.

Density	Cap	Refusal share	First invalidating step ↑	Seeds with an invalidation (of 4)	Invalidations per search ↓	Hours on later-refused ↓	Returned PPL ↓
low	0.40	0.000	no reading	0	0.00	0.000	7.05
low	0.22	0.000	no reading	0	0.00	0.000	8.45
low	0.14	0.000	no reading	0	0.00	0.000	9.69
low	0.09	0.015	13.00	1	0.25	4.000	11.68
medium	0.40	0.000	no reading	0	0.00	0.000	7.05
medium	0.22	0.125	1.00	1	0.25	0.800	8.43
medium	0.14	0.176	3.33	3	1.50	8.800	9.89
medium	0.09	0.167	3.75	4	3.25	24.800	12.36
high	0.40	0.000	no reading	0	0.00	0.000	7.05
high	0.22	0.400	1.00	2	2.00	8.800	8.92
high	0.14	0.306	2.25	4	3.75	21.600	9.97
high	0.09	0.234	5.75	4	3.75	22.400	12.36

Why the dictionary extreme makes the completion optimistic. The optimistic completion assigns an unobserved composition the most favourable correction the record already holds under the order u, and the zero correction stays that extreme until a correction with a higher u is observed, and such a correction entering the record moves the extreme rather than breaking it.

Why the loop terminates, and what it does when the witness set empties. Since every acquisition turn removes one composition from the unobserved set without returning it, and every build that does not halt records a composite refusal that withdraws a candidate, the loop is obtained as a strictly shrinking process that terminates after at most $|\mathcal{X}|(|\mathcal{X}| - 1)$ acquisition builds. When the two completions disagree but every ordered pair inside them has already been observed, W_t is empty while the decision is still open; the loop then draws its next observation from the unobserved pairs of the catalogue at large, and halts when there are none left.

APPENDIX A.4: PARAMETER TABLE

Every quantity this paper reports is produced by the PipeSearch loop evaluated on the composition model of Eq. 10 under the parameter tables below, and the tables are released with the code. No reading in this paper is measured on a served model; each is the value the composition model returns for the pipeline the loop selected.

The residual table ε of Eq. 10 is a further fixed parameter. It is keyed on the applied set the prefix before the last atom reached and on that atom, and it holds the gap between the readout a build returned and the readout the composition model predicted for it, written by Eq. 18 whenever a build

disagrees with its prediction. It is empty at the start of every run reported here and is written at most once, on the step that returns the pipeline, where no search reads it back. Throughout, $r_{\mathrm{obs}}(\sigma)$ is the value the composition model returns for the whole pipeline σ.

Apart from the middle motivation panel and Figure 5, every table and figure reproduces cell for cell, to the precision it is printed at, off the released code and catalogues, the largest of which holds 12 atoms over 132 ordered pairs. The 24-atom catalogue of Figure 5 is drawn for that figure alone from the six families of the setting grid at four settings each, and is not one of the released catalogues.

Two tables are needed because the arms read from two places. Table 12 is the per-family setting grid; a single-method arm is the best setting of its own family that clears the budget in question, or its most aggressive setting when none clears it, taken off that grid and quantized onto the decision grid of Eq. 9 before it is read. Each readout is formed by multiplying the solo ratios along the pipeline, adding one correction per ordered pair already in place, then clamping the two footprint components and the throughput component to at least 10^{-6} and the quality component to $[0, 1]$ and rounding each of the four components to the nearest multiple of 0.005. A catalogue is then drawn from those grids with a per-atom offset bounded by 0.012 in magnitude on the two footprint ratios and the throughput ratio and by a quarter of that on the quality ratio; Table 13 prints the catalogue the main tables were run on, after that offset, exactly as it is released in `catalogue_manifest.json`. Reproducing a single-method row needs the first table and reproducing a PipeSearch row needs the second.

One budget component is stated here rather than in the tables, because it applies to one figure alone. Figure 8 is read under an accelerator-time component of B set to $1 + 0.1\,K(K-1)$ hours, and every table and every other figure leaves that component unbinding; empty rows in the paper arise under the other components of B or under an arm's own label quota. Two consequences follow for the main tables. Their accelerator-time axis does not bind, and therefore the hours they report are what the loop spent and not what it was held to, and the common budget of Eq. 1 constrains them on the memory axis alone. Their PipeSearch rows are read on one label draw, seed 7, while the ablation and oracle tables pool several seeds, which means a quality reading in Table 2 and a pooled reading elsewhere are not the same statistic.

The ratios $r_w \downarrow$, $r_c \downarrow$, $r_t \uparrow$ and $r_q \uparrow$ are taken against the backbone's fp16 references, and the build cost is the accelerator-hour charge of one build at that setting.

The three tables below hold the released catalogue, the per-backbone references and the task offsets. The first is the catalogue: one row per atom, with the four readout ratios and its build charge.

The second holds the fp16 reference constants each ratio is taken against, one row per backbone.

Perplexity is recovered as $\mathrm{ppl}_{\mathrm{fp16}}(1 + \kappa(1 - r_q))$ with $\kappa = 1.85$ fixed across backbones, and the zero-shot average as the mean of the four task readings taken with the offsets of Table 15.

The third holds the link functions that map the quality axis onto the four zero-shot tasks.

The four tasks share one slope, $\mathrm{acc} = \mathrm{acc}_{\mathrm{fp16}} - \alpha(1 - r_q)$ with $\alpha = 22.0$ for all tasks and backbones; the offsets sum to zero, so the task columns average to the backbone's zero-shot link.

The serving columns are fixed functions of the footprint and throughput a pipeline reaches, and of nothing else: the peak footprint splits into a weight side $W_{\mathrm{fp16}}\, r_w$ and a KV side $52\, c_{\mathrm{kv}}\, r_c$ at 8192 tokens, where W_{fp16} is the backbone's fp16 weight memory and c_{kv} its KV cost per sequence, prefill latency reads $940\,000/T(\sigma)$ milliseconds with $T(\sigma)$ the decode throughput of Eq. 23 in tokens per second, and the sustained batch is what the device still holds once the weights are placed, $\lfloor (80 - W_{\mathrm{fp16}}\, r_w)/(c_{\mathrm{kv}}\, r_c) \rfloor$. No family enters these three links, and therefore the serving columns of Tables 1 and 3 move with the footprint and decode columns alone.

Each ordered pair carries either a refusal, at the fixed refusal density, or a correction on the four axes of the readout. The correction is drawn as a penalty: the composition reads worse on throughput and on quality than the two methods read apart, and worse on a footprint side only when both methods act on that side, since two methods that touch different pools cannot correct each other there. The one way a pair can read better on any axis is the order-asymmetric term, which scales the whole correction by $1 + \pi\varsigma$, where $\pi = +1$ when the first atom's name precedes the second's in alphabetical order and $\pi = -1$ otherwise, and ς is drawn for each ordered pair up to 0.60 times the order-gap level; the product turns negative, and the axes flip together, only where that level exceeds the point at which that skew can pass 1. Counted over the 154 distinct label tables the reported runs

Table 12: The per-family setting grid the single-method arms are read from. Setting 0 is the most aggressive.

Family	Setting	$r_w \downarrow$	$r_c \downarrow$	$r_t \uparrow$	$r_q \uparrow$	Build cost
weight_quant	0	0.3016	1.0000	1.2800	0.8850	1.600
weight_quant	1	0.4467	1.0000	1.1933	0.9233	1.183
weight_quant	2	0.5918	1.0000	1.1067	0.9617	0.767
weight_quant	3	0.7368	1.0000	1.0200	1.0000	0.350
kv_quant	0	1.0000	0.1250	1.4200	0.9300	1.600
kv_quant	1	1.0000	0.3410	1.2967	0.9533	1.183
kv_quant	2	1.0000	0.5571	1.1733	0.9767	0.767
kv_quant	3	1.0000	0.7731	1.0500	1.0000	0.350
evict	0	1.0000	0.2000	1.6100	0.8650	1.600
evict	1	1.0000	0.3836	1.4467	0.9100	1.183
evict	2	1.0000	0.5672	1.2833	0.9550	0.767
evict	3	1.0000	0.7508	1.1200	1.0000	0.350
width_slice	0	0.7000	0.7000	1.1900	0.9120	1.600
width_slice	1	0.7792	0.7792	1.1167	0.9413	1.183
width_slice	2	0.8583	0.8583	1.0433	0.9707	0.767
width_slice	3	0.9375	0.9375	0.9700	1.0000	0.350
lowrank	0	0.7500	1.0000	1.3300	0.8480	1.600
lowrank	1	0.8080	1.0000	1.2233	0.8987	1.183
lowrank	2	0.8661	1.0000	1.1167	0.9493	0.767
lowrank	3	0.9241	1.0000	1.0100	1.0000	0.350
unstructured	0	0.5618	1.0000	1.2400	0.8040	1.600
unstructured	1	0.6577	1.0000	1.1567	0.8693	1.183
unstructured	2	0.7535	1.0000	1.0733	0.9347	0.767
unstructured	3	0.8494	1.0000	0.9900	1.0000	0.350

Table 13: The released twelve-atom catalogue the main tables were run on, printed after the fixed per-atom offset.

Atom	Family	$r_w \downarrow$	$r_c \downarrow$	$r_t \uparrow$	$r_q \uparrow$	Build cost
weig0	weight_quant	0.3069	1.0000	1.2747	0.8863	1.600
kv_q0	kv_quant	0.9881	0.1131	1.4319	0.9270	1.600
evic0	evict	0.9984	0.1984	1.6116	0.8646	1.600
widt0	width_slice	0.6927	0.6927	1.1973	0.9102	1.600
lowr0	lowrank	0.7615	1.0000	1.3185	0.8509	1.600
unst0	unstructured	0.5621	1.0000	1.2397	0.8041	1.600
weig1	weight_quant	0.4488	1.0000	1.1912	0.9238	1.183
kv_q1	kv_quant	0.9991	0.3401	1.2976	0.9531	1.183
evic1	evict	1.0000	0.3910	1.4393	0.9119	1.183
widt1	width_slice	0.7767	0.7767	1.1192	0.9407	1.183
lowr1	lowrank	0.8031	0.9951	1.2282	0.8975	1.183
unst1	unstructured	0.6479	0.9902	1.1665	0.8668	1.183

draw, one per catalogue, seed, refusal density and order gap, 37 of the 5361 ordered pairs that are not refusals carry a favourable axis, a share of 0.0069, all of them in the large-order-gap stratum. This is what makes the optimistic completion of Eq. 5 something other than the zero correction.

APPENDIX A.5: SEARCH PROCEDURE AND EXACTNESS SCOPE

Why quantization belongs to the state. Quantizing ρ is part of the state and not an approximation applied to it, which means two prefixes are the same state exactly when the three fields agree and the transition becomes a function of the state alone.

Table 14: Reference constants per backbone. The weight column is the parameter count at two bytes and the cache column is what one sequence of 8192 tokens holds, both read off the released architecture; the footprint of a deployment is the first plus the concurrency times the second.

Backbone	fp16 weights (GB)	KV per seq. (GB)	Thr. ratio	fp16 PPL	fp16 zero-shot
Llama-3.1-8B	16.00	1.074	1.000	6.24	69.7
Qwen2.5-7B	15.22	0.470	1.045	6.85	70.4

Table 15: Zero-shot task links. A task's reading is the backbone's zero-shot average plus the offset below, at every level of retained quality.

Task	fp16 offset
ARC	-8.4
PIQA	+9.1
HellaSwag	-1.7
WinoGrande	+1.0

Why the merge is exact. The merge is exact because the transition is a function of the state. Two readouts are compared on total footprint, throughput and quality, with the footprint negated so that the order runs higher-is-better throughout, and dominance discards a readout no better under $\trianglelefteq$.

Why the per-slot multiplier grows. The multiplier grows because the j-th future atom meets $|A(s)| + j - 1$ atoms already in place and owes that many corrections; a fixed multiplier of $|A(s)|$ would under-count them and could prune the true optimum, and therefore we set the slot-growing form, which yields $h(s) \geq V(s)$ under any labelling drawn from $\mathcal{D}_t$ and makes best-first pruning discard nothing that could win under that labelling.

This is a combinatorial dominance argument over a finite dictionary rather than a probabilistic bound, and it is relative to the composition model of Eq. 10 under one fixed residual table. It is also relative to the order u of Eq. 5: u is realised as a weighted sum of the four axes with fixed weights, a total order on corrections rather than the componentwise partial order; therefore a correction the componentwise order leaves incomparable is still ranked, and the optimistic completion is optimistic with respect to that ranking and not to every axis at once. What this component obtains is the pipeline the observed structure implies, with the ordering of Eq. 1 preserved throughout.

The record is deliberately coarser than the event that produced it: every order over the same atoms that reaches the same applied set inherits a composite refusal even though the failure was observed for one of them, and this over-blocking can only withdraw candidates and never admit one, which means no returned pipeline rests on an unobserved interaction.

Algorithm 1 summarises the loop that composes the three components, with the turn subscript dropped inside the box so that every quantity is read at its current value. Each turn first runs the order-aware prefix search once under the optimistic completion and once under the pessimistic one, expanding the frontier best-first under the order-correct bound, testing every successor against the budget predicate of feasibility maintenance, merging it into its state class, and discarding it when a previously retained state dominates it. The incumbent of a search is the best prefix admitted so far whose readout fits the budget, and the prefix it holds when the bound can no longer beat it is the pipeline that completion implies. Decision-relevant acquisition then compares the two pipelines and, when they differ, observes the cheapest witness and charges its build to the ledger; when they agree, the agreed pipeline is built and read, and the reading is recorded, forbidding its last atom at the applied set its prefix reaches when it misses the budget and writing the residual otherwise. The loop returns once the three conjuncts of the halting condition hold, or returns the best entitled optimistic pipeline reached so far, or none, when an axis of the budget runs out first.

The cost of one such loop is stated in Table 16 over the state abstraction of Eq. 9. All quantities there are host-side except the builds, which are the only operations debited to the accelerator-time axis of B: the table separates what the search costs to run from what it costs to observe.

Algorithm 1 PipeSearch

Require: catalogue $\mathcal{X}$, budget B, length cap $L_{\max}$
Ensure: pipeline $\sigma^\star$ and the observation record $\mathcal{M}$ that entitles it

read the given solo readouts r_{solo}; $\mathcal{M} \leftarrow \varnothing$; $\mathcal{D} \leftarrow \{\bot, \mathbf{0}\}$; $\Psi \leftarrow \varnothing$; $\varepsilon \leftarrow \varnothing$; $C^{\text{acq}} \leftarrow 0$
repeat
Part 1: order-aware prefix search, once under each extreme completion
  for $\lambda \in \{\lambda^+, \lambda^-\}$ **do** ▷ Eq. 5
    frontier $\leftarrow$ {the empty prefix, forbidding $\Psi(\varnothing)$}; $\hat{\sigma} \leftarrow$ empty prefix **if** $\rho(\varnothing) \preceq B$, **else** unset
    while the frontier holds a state s with $h(s) > Q(\rho(\hat{\sigma}))$, or any s while $\hat{\sigma}$ is unset **do** ▷ Eq. 15
      pop the best-first such s
      for each $x \notin F(s)$ of a family new to $A(s)$, with $|A(s)| < L_{\max}$ **do**
        $s' \leftarrow s \oplus x$ ▷ Eq. 11
        if $\Xi(s') = 0$ **then continue** ▷ feasibility maintenance, Eq. 16
        if a previously retained $s'' \equiv s'$ exists **then** merge s' into it, keeping the prefix already there, and **continue** ▷ Eq. 12
        if $s' \prec s''$ for a previously retained state s'' **then continue** ▷ Eq. 13
        push s'; $\hat{\sigma} \leftarrow s'$ **if** $\rho(s') \preceq B$ and ($\hat{\sigma}$ is unset or $Q(\rho(s')) > Q(\rho(\hat{\sigma}))$)
      end for
    end while
    $\sigma^\lambda \leftarrow \hat{\sigma}$
  end for
Part 2: decision-relevant acquisition, or the build it has earned
  if $\sigma^+ \neq \sigma^-$ **then**
    $(x \to x') \leftarrow \arg\min_{(x \to x') \in W} c(x \to x')$, or the cheapest unobserved pair of distinct atoms of $\mathcal{X}$ when $W = \varnothing$ ▷ Eq. 8
    if $C^{\text{acq}} + c(x \to x')$ fits the accelerator-time axis of B **then** observe it; add it to $\mathcal{M}$ and $\mathcal{D}$; $C^{\text{acq}} \leftarrow C^{\text{acq}} + c(x \to x')$; **else** that axis is exhausted
  else if σ^+ is unset **then**
    return the best entitled optimistic pipeline reached so far, or none, and $\mathcal{M}$
  else
    $\sigma^\star \leftarrow \sigma^+$; build it and read $r_{\text{obs}}(\sigma^\star)$; add $\sigma^\star \mapsto r_{\text{obs}}(\sigma^\star)$ to $\mathcal{M}$ ▷ feasibility maintenance, Eq. 18
    if $r_{\text{obs}}(\sigma^\star) \not\preceq B$ **then** forbid $\text{last}(\sigma^\star)$ at $A_{\sigma^\star}$ **else** write the residual $\varepsilon(A_{\sigma^\star}, \text{last}(\sigma^\star))$
  end if
  if an axis of B is exhausted before the halting condition holds **then** $\sigma^\star \leftarrow$ the best entitled optimistic pipeline reached so far, or none
until the three conjuncts of the halting condition hold, or an axis of B is exhausted ▷ Eq. 19
return $\sigma^\star$ and $\mathcal{M}$, and on exhaustion the axis that ran out

Table 16: Search cost of one loop, stated over the state abstraction of Eq. 9. All quantities except the builds are host-side, and only builds are debited to the accelerator-time axis of B.

Quantity	Growth
Reachable states	polynomial in the per-family setting grid
Acquisition builds to halt	at most the ordered-pair count
Searches to halt	$2\,(\lvert\mathcal{X}\rvert(\lvert\mathcal{X}\rvert - 1) + R + 1)$, with R the composite refusals

There $\mathcal{K}$ is the set of method families, Θ_k the setting grid of family k, $\mathcal{X}$ the catalogue of atoms (one family at one setting), and G the number of distinct forbidden sets reachable under the observed refusals. The count of $(A(s), F(s))$ keys is bounded by $\big(\prod_{k \in \mathcal{K}} (|\Theta_k| + 1)\big) \cdot |\mathcal{K}| \cdot G$, and the reachable-state count by that product times the quantized readouts the orders over one applied set produce; acquisition builds to halt are bounded by $|\mathcal{X}|(|\mathcal{X}| - 1)$ and searches to halt by $2\,(|\mathcal{X}|(|\mathcal{X}| - 1) + R + 1)$ with R the composite refusals recorded.

Realized Search Cost. Table 16 states the growth of one loop over the state abstraction of Eq. 9, and the runs place that growth on the charged axis. Acquisition builds to halt are bounded by the ordered-pair count, and the loop spends 0.201 to 0.500 of it: the realized acquisition cost is 19.2 accelerator-hours at $K{=}4$ and 75.1 at $K{=}12$ against 38.4 and 367.4 for complete labelling, a ratio that widens from 2.0 to 4.9 as the catalogue grows. Paying for every candidate instead of the witness set costs 96.0 hours at both memory budgets, which is where the witness rule shows up as a cost and not as a readout.

APPENDIX A.6: FEASIBILITY AND INTEGRATION DETAILS

Why the budget subtraction touches one axis only. Only pairwise acquisitions are added to C^{acq}; the build of an agreed pipeline is not an acquisition, and every search cost reported here counts pairwise builds alone. The acquisition spend C_t^{acq} is a vector supported on the accelerator-time axis of B alone; therefore the subtraction in Eq. 16 is a subtraction on that axis and the identity on device memory and sustained throughput, which a finished trial build neither holds nor withdraws.

What refusal propagation removes from the frontier. The bound $h(s)$ of Eq. 15 is evaluated only on states none of whose applied ordered pairs has been refused under the current labelling, which means a refusal arriving at turn t removes every frontier state that applied the refused pair.

How a readout is compared to the budget. We write $\rho(s) \preceq B$ and $r_{\mathrm{obs}}(\sigma) \preceq B$ for the deployment cost that readout implies fitting the budget on the axes of B, since the readout is a ratio tuple and B is a cost vector.

Why the composite record is safe. Both branches are written on the pair (applied set, last atom) rather than on the full state, which means they do not distinguish two states that differ only in their readout or their forbidden set; the record lookup depends on that key alone while the transition stays a function of the full state, and the merge of Eq. 12 stands. The two branches are exclusive, and the second fires only on the step that returns the pipeline, which is the step the loop stops on, and therefore a residual written there is never read back by the search that wrote it.

Framework Summary. PipeSearch makes single-accelerator compression one loop that pays for structure discovery in the currency of deployment: acquisition buys labels only while the completions disagree, the search returns the sequence they imply, and feasibility keeps it inside the budget.

What the loop returns when a budget axis runs out. When an axis of B is exhausted before the three conjuncts of the halting condition hold, we return the best entitled optimistic pipeline reached so far together with the axis that ran out, since entitlement is checked per candidate.

APPENDIX A.7: SOFTWARE AND DETERMINISM

Every reading in this paper is produced by one Python entry point, run from inside the released code directory with every path relative to it. Nothing outside the Python standard library is imported except the plotting package pinned in the requirements file, which is used only to draw the figures and takes no part in producing a reading.

A run is deterministic and repeats exactly. Every ordering the loop performs is an explicit sort rather than a dictionary or set traversal, and every random draw comes from a generator seeded from one fixed base seed, with the catalogue of size K drawn from that seed and K together. Re-running the released code reproduces every number reported here bit for bit.

Accelerator time is charged from the parameter table rather than timed on a device, which is what lets the loop be replayed on any host that runs the interpreter and makes every hour it reports independent of the machine a reader has.

Table 17: What the budget grid spans, from a cap a single method clears to one that takes four.

Budget (fraction of fp16 peak)	0.40	0.32	0.22	0.14	0.09
Single-method families clearing the cap	2	1	0	0	0
Methods in the returned pipeline	1	1	2	3	4

APPENDIX A.8: BUDGET GRID

The loosest cap is fixed by the catalogue. The tightest memory ratio any method family reaches on its own setting grid is 0.320, which means at any cap under that floor no single-method arm clears the budget and every single-method column reads its most aggressive setting over the cap. A cap just above the floor does not fix this either: at 0.32 exactly one family clears, therefore the single-method regime exists but rests on one arm. We therefore set the loosest cap where that regime is non-degenerate, at 0.40, where two families clear on their own; 0.40 is also the loose end of the memory axis Figure 7 is read on. The three tighter caps each tighten the one before by a factor between 0.55 and 0.64, which are ratios of the caps themselves rather than readings, carried down until four methods are needed.

Table 17 reads what the grid spans: the number of families that clear each cap on their own and the number of methods in the full loop's pipeline, with the 0.32 corner carried alongside the four caps.

Its first row counts the families whose setting grid reaches the cap on Llama-3.1-8B, and the second the methods in the pipeline PipeSearch returns at that cap in the main-table setting.

At the loosest cap the grid does not separate the arms: at 0.40 the loop returns a single method, the lesioned arms read the perplexity the full loop reads, and the ablation column carries no difference.

APPENDIX A.9: EVALUATION METRICS

Five readings are reported. Each is the value the composition model returns for the pipeline the loop selected, taken at the quantized readout ρ of Eq. 9 rather than measured on a served model.

Perplexity readout. The quality axis r_q of a pipeline is mapped to a perplexity reading against the backbone's fp16 reference:

$$\mathrm{PPL}(\sigma) \;=\; \mathrm{PPL}_{\mathrm{fp16}}\big(1+\kappa\,(1-r_q(\sigma))\big), \tag{21}$$

where $\kappa = 1.85$ is the slope set in Appendix A.4. A pipeline that retains all quality therefore reads the reference and one that loses quality reads above it. Lower is better.

Zero-shot average. Each task reads the backbone's fp16 zero-shot accuracy reduced in proportion to the quality lost, from that task's fp16 reading:

$$\mathrm{acc}_k(\sigma) \;=\; \mathrm{acc}_k^{\mathrm{fp16}} \;-\; \alpha\big(1-r_q(\sigma)\big), \tag{22}$$

with one slope α shared across tasks and backbones. The reported average is the mean over the four tasks. Higher is better.

Peak device memory and decode throughput. Both are read off the remaining axes of ρ against the backbone's fp16 references:

$$M(\sigma) \;=\; W_{\mathrm{fp16}}\, r_w(\sigma) \;+\; N_{\mathrm{ref}}\, c_{\mathrm{kv}}\, r_c(\sigma), \qquad T(\sigma) \;=\; r_t(\sigma)\, T_{\mathrm{fp16}}, \tag{23}$$

where W_{fp16} is the backbone's fp16 weight memory, c_{kv} its cache cost per sequence at the 8192-token context, N_{ref} the reference batch of 52 sequences, and T_{fp16} its throughput reference, 1000 tokens per second times the backbone's throughput ratio. The two sides are added because they are separate pools: what a method does to one pool is not offset by what it does to the other. Memory is lower-better and throughput higher-better.

Table 18: Deleting the order field of the search state, by order-gap stratum and memory budget.

Arm	Order gap	Cap	PPL gap to best reachable ↓	Zero-shot gap to best reachable ↓	Peak mem. (GB) ↓
PipeSearch (full)	**near-zero order gap**	**0.40**	**0.000**	**0.000**	**23.26**
PipeSearch (full)	**near-zero order gap**	**0.22**	**0.000**	**0.000**	**12.06**
PipeSearch (full)	**medium order gap**	**0.40**	**0.000**	**0.000**	**23.26**
PipeSearch (full)	**medium order gap**	**0.22**	**0.000**	**0.000**	**12.06**
PipeSearch (full)	**large order gap**	**0.40**	**0.000**	**0.000**	**23.26**
PipeSearch (full)	**large order gap**	**0.22**	**0.000**	**0.000**	**12.06**
Order-collapsed search (directions pooled)	near-zero order gap	0.40	0.000	0.000	23.26
Order-collapsed search (directions pooled)	near-zero order gap	0.22	0.224	0.426	12.06
Order-collapsed search (directions pooled)	medium order gap	0.40	0.000	0.000	23.26
Order-collapsed search (directions pooled)	medium order gap	0.22	0.079	0.151	12.06
Order-collapsed search (directions pooled)	large order gap	0.40	0.000	0.000	23.26
Order-collapsed search (directions pooled)	large order gap	0.22	0.036	0.069	12.06

Acquisition fraction. The label cost of a run is reported as the share of the ordered pairs the catalogue admits that the loop actually built:

$$\mathrm{AcqFrac} \;=\; \frac{\left|\{(x \to x') \in \mathcal{M}_{\mathrm{end}}\}\right|}{K(K-1)}, \tag{24}$$

where the numerator counts the ordered pairs in the observation body $\mathcal{M}_{\mathrm{end}}$ the loop holds when it stops, and not the composite record of the returned pipeline. Lower is better.

Pooling. Every ratio reported in this paper is a pooled count: one numerator over one denominator across the cells it covers. A mean of per-cell ratios would weight a cell with few builds as heavily as one with many, and therefore it is never used.

APPENDIX A.10: ORDER-COLLAPSE AND FEASIBILITY ABLATIONS

This part deletes the order field of the search state, stratified by order gap, and moves refusal handling out of expansion time.

Order Lesion by Order Gap. Table 18 stratifies the order lesion by order gap. At the 0.40 cap the collapsed arm reaches the best reachable pipeline in every stratum, with a gap of 0.000 on both columns. At the 0.22 cap its perplexity gap reads 0.224 in the near-zero stratum, 0.079 in the medium stratum and 0.036 in the large stratum, with zero-shot gaps of 0.426, 0.151 and 0.069, against 0.000 for the full loop throughout. Both of its gap columns are read against the best pipeline reachable under complete labelling on the same seed, and each arm is read on the seeds on which it returned a pipeline and the reference was reachable.

The second table separates where refusals are handled. It compares excluding a refused composition at expansion time against filtering the finished list, at each refusal density.

The first column counts expansions attempted into a region an observed refusal has already forbidden, not expansions a future refusal will later invalidate. The comparison the table carries is therefore the expansion counts rather than the returned perplexity.

Placement of Refusal Handling. Table 19 moves refusal handling out of expansion time. The two arms return the same perplexity readout at every refusal density, 10.07 at low, 9.96 at medium and 10.47 at high. What separates them is the search that gets there. At high density expansion-time

Table 19: Where refusals are handled, by refusal density: expansions into the forbidden region ↓, states expanded ↓, and returned perplexity ↓.

Arm	Refusal density	Expansions into forbidden region ↓	States expanded ↓	Final PPL ↓
expansion-time exclusion (full)	**low**	**0.0**	**6631.9**	**10.07**
expansion-time exclusion (full)	**medium**	**0.0**	**6886.5**	**9.96**
expansion-time exclusion (full)	**high**	**0.0**	**6553.9**	**10.47**
post-hoc filtering	low	710.9	7342.8	10.07
post-hoc filtering	medium	652.4	7538.9	9.96
post-hoc filtering	high	1841.4	8395.2	10.47

Table 20: How far the lesioned arms sit from the full loop, by memory cap, on the six-atom catalogue, pooled over four seeds.

Setting	Full loop PPL ↓	Largest lesion gap	Mean lesion gap	Seeds returning (of 4)
cap=0.40	7.048	0.000	0.000	4
cap=0.32	8.246	0.144	0.067	4
cap=0.22	8.419	0.058	0.019	4
cap=0.14	9.876	0.750	0.269	4

exclusion expands 0.0 states into the forbidden region out of 6553.9 expanded in total, while post-hoc filtering expands 1841.4 such states out of 8395.2. The first column is zero for the full arm by construction: it withdraws a forbidden atom before a successor is built.

The third table reads how far each lesioned arm sits from the full loop at the two caps the lesion table is read on, at the loosest cap and at the 0.32 corner, which is what decides whether a cap separates the arms at all.

It reads the full loop's perplexity ↓, then the largest and the mean absolute perplexity gap to it over the three lesion arms, and how many of the four seeds returned a pipeline.

APPENDIX A.11: DETAILED READINGS OF THE EXPERIMENTAL TABLES AND FIGURES

This section reads each table and figure of the experiments in full, including the reading conventions the captions leave out, and carries the oracle comparison the body points to.

Memory Fit. Table 1 is read at 8192 tokens and 52 concurrent sequences on the two backbones that carry footprint readings. The weight-side column is the parameter count at two bytes scaled by what the pipeline does to the weights, the KV-side column is the cache those 52 sequences hold at that context scaled by what the pipeline does to the cache, and the device figure is their sum. The share each side holds therefore follows from the architecture and moves with the context and the concurrency rather than being fixed. The two backbones are read side by side; therefore a row is one setting and its two triples are that setting on Llama-3.1-8B and on Qwen2.5-7B.

Served Quality. Table 2 is read under a memory cap of $0.14\times$ the fp16 footprint. The Avg. column is the mean of the four zero-shot accuracies and is the backbone's zero-shot link rather than a fifth reading; the quality columns are obtained through the quality links. The rows are not read at the same peak memory, and Table 1 records the footprint each arm reaches.

Serving Readings. Table 3 carries four columns that all follow from the serving links: prefill latency is 940.0 divided by the decode throughput in thousands of tokens per second, and the sustained batch is what the device still holds once the weights are placed; both are fixed once the decode column and the two sides of the footprint are. As in Table 2, the rows differ in realized peak memory.

Table 21: The returned pipeline against complete offline labelling on the same seeds: identity, the two gaps, the fraction of ordered pairs built, and search cost.

Arm	K	Identity to oracle ↑	PPL gap to oracle ↓	Zero-shot gap to oracle ↓	Fraction of $K(K-1)$ built ↓	Search cost (h) ↓
PipeSearch (full)	**4**	**1.000**	**0.000**	**0.000**	**0.500**	**19.200**
Order-collapsed search (directions pooled)	4	0.000	0.539	1.027	0.813	31.200
Complete offline labelling oracle	4	1.000	0.000	0.000	1.000	38.400
PipeSearch (full)	**6**	**1.000**	**0.000**	**0.000**	**0.275**	**26.400**
Order-collapsed search (directions pooled)	6	0.500	0.750	1.430	0.967	92.800
Complete offline labelling oracle	6	1.000	0.000	0.000	1.000	96.000
PipeSearch (full)	**8**	**1.000**	**0.000**	**0.000**	**0.259**	**43.064**
Order-collapsed search (directions pooled)	8	0.000	0.346	0.660	0.844	139.524
Complete offline labelling oracle	8	1.000	0.000	0.000	1.000	167.524
PipeSearch (full)	**12**	**1.000**	**0.000**	**0.000**	**0.201**	**75.105**
Order-collapsed search (directions pooled)	12	0.250	0.202	0.385	0.818	296.603
Complete offline labelling oracle	12	1.000	0.000	0.000	1.000	367.356

Component Ablation. In Table 6 each reading column is the mean over the seeds on which that arm returned a pipeline, which means the arms are read on their own sets of returning seeds rather than on one common set; the label tables of all four seeds are released in `catalogue_manifest.json`, from which a pooled row can be recomputed.

Comparison with the Oracle. In Table 21 the identity column is a pooled count over seeds, and the two gap columns are taken over the seeds on which that arm returned a pipeline and the oracle reached one. The fraction built counts the $K(K-1)$ ordered pairs actually observed, and the search-cost column carries the accelerator-hours those observations were charged.

The arm that pools the two directions of a pair falls short of the oracle at every catalogue size. Its perplexity gap to the oracle is 0.539 at $K = 4$, 0.750 at $K = 6$, 0.346 at $K = 8$ and 0.202 at $K = 12$, and its identity to the oracle reads 0.000, 0.500, 0.000 and 0.250 over the same range, while PipeSearch reads 1.000 identity and a zero gap throughout. On cost the pooled arm is the dearer of the two at every size: 31.2 accelerator-hours against 19.2 at $K = 4$, 92.8 against 26.4 at $K = 6$, 139.5 against 43.1 at $K = 8$, and 296.6 against 75.1 at $K = 12$, where both sit under the 367.4 hours complete labelling charges for the same catalogue. Pooling the two directions therefore buys neither a cheaper search nor the oracle's pipeline: the fraction it builds runs from 0.813 to 0.967, where PipeSearch stays between 0.201 and 0.500.

Acquisition Ledger. The three lesioned arms are named by what they remove: the order-aware search gives way to a state without its order field over a readout that pools the two directions of a pair, the agreement-stop arm replaces the stop with a quota of a tenth of the $K(K-1)$ pairs and returns its optimistic pipeline without checking the built readout against B, and the witness-rule arm pays for every candidate. In Table 5 hours per returned pipeline is the acquisition hours divided by the pipelines returned, both pooled over the four seeds, and the host-side state count is charged to neither the memory nor the throughput axis of B. The arm with the agreement stop dropped expands 2970.2 host-side states against 12695.2 at the 0.09 cap and pays 9.600 hours per returned pipeline against 6.400 at the looser budget.

Quality across the Memory Grid. Figure 7 is read through Eq. 10 under the parameter table on a grid of eight memory points. The horizontal axis is the memory budget as a fraction of the fp16

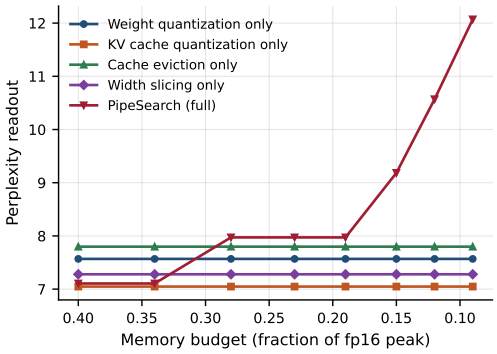

Figure 7: Served quality against the memory budget for each single-method arm and for PipeSearch. The axis runs loose on the left to tight on the right.

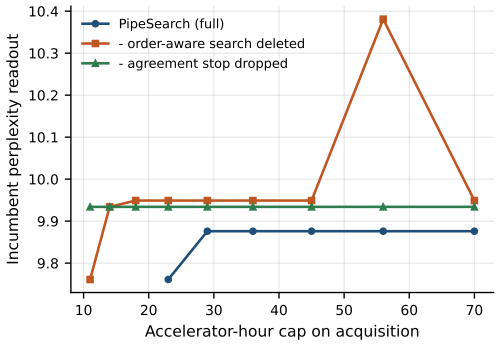

Figure 8: The incumbent perplexity ↓ each arm returns under a cap on acquisition accelerator-hours, at the main memory cap. Each point is a separate run.

peak and runs loose on the left to tight on the right, which means moving right is a constraint being tightened.

Figure 7 sweeps the single-method arms and PipeSearch across a memory grid of eight points, which runs from 0.40 down to 0.09 and spans the four caps. Only PipeSearch is composed anew at each point, and its perplexity is non-decreasing over the grid with two stretches where a tightening costs it nothing, while a single-method arm below the budget its most aggressive setting reaches is read at that setting rather than inside the budget. The composition is what supplies the extra points: a family exhausts its own grid, and a second family carries the tightening the first cannot.

Incumbent under an Hour Cap. Figure 8 reads the incumbent perplexity at the main memory cap against a cap on the accelerator-hours charged to acquisition under Eq. 20. Each point is a separate run held to its own cap rather than a position along one accumulating trajectory; therefore the seeds that returned a pipeline can differ between points. A cap at which no seed returned carries no marker, and those are the six tightest caps of the full loop and the three tightest of each of the other two arms; no curve has an interior gap.

The arm with the order-aware search deleted first reads 9.761 at a cap of 11.0 accelerator-hours and the arm with the agreement stop dropped first reads 9.934 there, while the full loop's first reading arrives at 23.0. By a cap of 70.0 accelerator-hours the full loop's incumbent reads 9.876 perplexity against 9.949 for the arm with the order-aware search deleted, and the arm with the agreement stop dropped holds 9.934 from 11.0 accelerator-hours onward.

APPENDIX A.12: DISCUSSION

The readings describe the loop under the parameter table. The search is rebuilt from the empty prefix on every outer iteration, carrying the observations, residual table and composite-refusal record forward. The refusal density, the order asymmetry and the interaction corrections are properties of that parameter table: what is tested is the loop's behaviour under a structure of that shape and not how often that shape arises. The entitlement result stands on the record the loop keeps: every ordered pair applied in a returned pipeline was one the loop paid to observe.

Where the accelerator-hour cap is exhausted before the agreement stop fires the loop returns the best entitled optimistic pipeline it reached, or none where nothing was entitled by then.