RKSC: Reasoning-Aware KV Cache Sharing and Confident Early Exit for Multi-Step LLM Inference

Anirudh Sekar

Abstract

We introduce **RKSC** (Reasoning-Aware KV Cache Sharing), a training-free inference framework that eliminates two structural redundancies in multi-branch LLM reasoning pipelines. **ASKS** (Attention-Similarity KV Sharing) computes the prefix KV cache once and broadcasts it to all semantically similar branches via hidden-state cosine similarity, strictly generalising the token-exact prefix caching used by vLLM and SGLang. **CGEE** (Confidence-Gated Early Exit) applies two complementary exit mechanisms: (1) it skips the verification forward pass entirely when generation confidence is decisive across branches, and (2) it terminates the verification pass at an intermediate layer when per-layer entropy stabilises, using lightweight hooks on the transformer backbone. **RSBCM** (Reasoning-Selective Block Cache Manager) prevents unbounded cache growth via attention-weighted depth-priority eviction. Across five model families (7B–10B), four benchmarks, and 1,000 evaluated problems, RKSC achieves a mean speedup of $\mathbf{3.008}\times$ over the No-KV baseline (peak $\mathbf{3.990}\times$), a $\mathbf{1.66}\times$ mean improvement over vLLM-equivalent prefix caching, with a CGEE-induced error rate of only 0.37% (6 errors out of 1,616 verify calls). No fine-tuning or architecture changes are required. Code is available at `https://github.com/AnirudhSekar/RKSC`.

1. Introduction

Inference-time scaling systems, DeepSeek-R1 (Guo et al., 2025), Qwen3 (An Yang et al., 2025), o1-style models, generate multiple reasoning trajectories per problem, with the first trajectory correct in the majority of cases (Chen et al., 2025b). This multi-branch paradigm incurs two structural redundancies that no production system addresses. **(1) Cross-branch KV waste:** parallel branches share the majority of their prefix KV computation yet recompute it independently, in a Tree of Thoughts (Yao et al., 2023) run (branching 4, depth 3), branches diverging at depth 2 share 66% of KV work by construction. vLLM (Kwon et al., 2023) and SGLang (Zheng et al., 2024) reuse KV blocks on exact token matches only, leaving all cross-branch similarity unexploited. **(2) Over-deep verification:** process-reward verification (Lightman et al., 2023) runs full-depth forward passes even when the model is already confident; no system exploits per-layer entropy collapse to exit early *within* a single verify pass.

We introduce **RKSC** (Reasoning-Aware KV Cache Sharing), a training-free inference framework eliminating both redundancies. Our contributions are:

- **ASKS** (§2.2): gates prefix KV reuse across branches via *hidden-state cosine similarity* with exponentially weighted later-layer emphasis, strictly generalising token-exact caching. Achieves 28.6% additional reuse where token-exact matching yields 0%.
- **CGEE** (§2.3): dual-level exit combining (a) *verify-skip* when generation confidence is decisive, and (b) *layer-level entropy exit* terminating the verify pass when per-layer logit entropy stabilises.

KV caching (Vaswani et al., 2023) and block-level prefix reuse (vLLM (Kwon et al., 2023), SGLang (Zheng et al., 2024)) require byte-identical token prefixes; ASKS decouples reuse from lexical identity. MemShare (Chen et al., 2025a) deduplicates KV states *within* a single chain via step-delimiter segmentation; ASKS operates *across* parallel branches. Layer-wise early exit has been applied to BERT (Xin et al., 2020) and generation (Schuster et al., 2022; Elhoushi et al., 2024); for reasoning, (Yang et al., 2025) exit *between* chains at transition tokens, a token-level mechanism orthogonal to CGEE's within-pass layer-level exit. Self-consistency (Wang et al., 2023) and process-reward methods (Lightman et al., 2023) motivate multi-branch generation; RKSC accelerates any such pipeline as a drop-in wrapper, without altering the reasoning algorithm.

. Correspondence to: Anirudh Sekar <anirudhsekar2008@gmail.com>.

Accepted to the ICML 2026 Workshop on Statistical Frameworks for Uncertainty in Agentic Systems, Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

A complementary line of inference acceleration uses a small draft model to propose candidate tokens, which a larger target model then verifies in parallel. (Leviathan et al., 2023) establish that this approach preserves the output distribution exactly while achieving 2–3× wall-clock speedup, and SpecInfer (Miao et al., 2024) generalises it to tree-structured candidate sets verified in a single batched forward pass. Medusa (Cai et al., 2024) removes the need for a separate draft model entirely by appending lightweight decoding heads to the backbone, predicting multiple future tokens in parallel and verifying them via tree attention. RKSC's CGEE is complementary to all three: whereas speculative methods accelerate the decode phase by parallelising token proposals, CGEE accelerates the verification phase of process-reward scoring by gating or truncating the verify forward pass based on per-layer entropy collapse.

2. RKSC Pipeline

RKSC accelerates multi-branch reasoning inference through three complementary mechanisms: (1) **KV prefix sharing** eliminates redundant prefill computation across branches, (2) **CGEE** reduces or eliminates the verification forward pass, and (3) **RSBCM** manages cache capacity under deep tree search. Figure 1 illustrates the full pipeline. The design exploits structure that already exists in the inference workload, shared prefixes, concentrated generation confidence, entropy collapse in later layers, rather than introducing new computation. Each mechanism is independently switchable and adds zero learnable parameters.

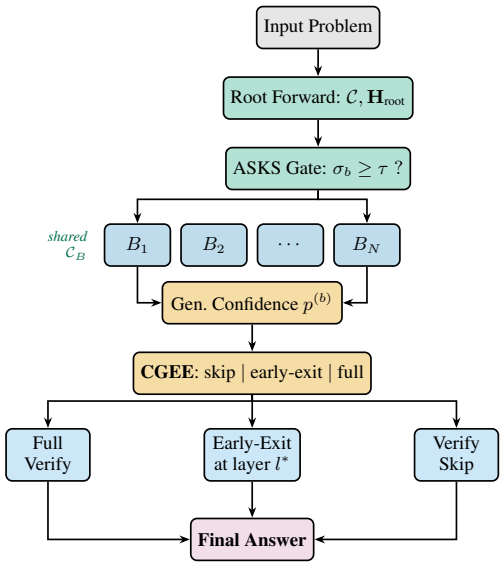

Figure 1. **RKSC pipeline overview.** A single root forward computes the shared KV cache $\mathcal{C}$ and root hidden states. ASKS broadcasts $\mathcal{C}$ only to branches with similarity $\sigma_b \geq \tau$. All B branches decode in one batched call while accumulating generation confidence. CGEE selects among three paths: *verify skip* (confidence decisive), *early-exit verify* (entropy stabilised at l^*), or *full verify*.

2.1. Problem Setting

We consider an LLM f_θ with L transformer layers solving a problem x by generating B reasoning branches from a shared problem prefix. Each branch $b \in \{1, \ldots, B\}$ begins from a shared context $\mathbf{c} = \texttt{prefix}(x)$ of n tokens and appends a short, branch-specific suffix (a reasoning hint of $s \ll n$ tokens) before decoding t answer tokens autoregressively. A *verify* pass then scores all B branches and selects the best answer. The standard inference procedure, which we call the *No-KV* baseline, batches all B branches in a single forward pass but recomputes the full n-token prefix attention independently for each branch, paying $O(Bn^2)$ attention cost. A stronger baseline, the *vLLM-equivalent*, performs token-exact prefix matching: the root KV cache is reused only when branch tokens share a byte-identical prefix with the root, mirroring the PagedAttention prefix-cache path in vLLM (Kwon et al., 2023) and SGLang (Zheng et al., 2024). This baseline captures all gains that existing production systems achieve and serves as our primary comparison point.

Algorithm 1 presents the top-level RKSC solve procedure. Detailed per-mechanism pseudocode appears in Appendix B.

Algorithm 1 RKSC top-level solve

Require: prefix $\mathbf{c}$, suffixes $\{\mathbf{s}_b\}_{b=1}^{B}$, thresholds $(\tau, \tau_{\text{conf}}, r_{\text{gap}}, \theta, \epsilon)$

$\mathcal{C}, \mathbf{H}_{\text{root}} \leftarrow f_\theta^{\text{prefill}}(\mathbf{c})$ {single root forward (§2.2)}
sharedSet $\leftarrow$ ASKSGATE$(\mathbf{H}_{\text{root}}, \{\mathbf{s}_b\}, \tau)$ {Alg. 2}
$\mathcal{C}_B \leftarrow \texttt{repeat_interleave}(\mathcal{C}, B)$; $\mathcal{C}_B$.clone()
$\{\mathbf{y}_b\}, \{p^{(b)}\} \leftarrow$ BATCHEDDECODE$(\{\mathbf{s}_b\}, \mathcal{C}_B)$ {track top-1 probs}
if $\max_b p^{(b)} \geq \tau_{\text{conf}}$ **and** gap-ratio $\geq r_{\text{gap}}$ **then**
  return $\arg\max_b p^{(b)}$ {verify-skip, Level 1}
end if
return VERIFYWITHENTROPYEXIT$(\{\mathbf{y}_b\}, \theta, \epsilon)$ {Alg. 3, Level 2}

2.2. ASKS: Attention-Similarity KV Sharing

Prefix forward pass. RKSC begins each solve call with a single prefix forward pass through all L layers, storing the resulting KV states and hidden representations:

$$\mathbf{K}^{(l)}, \mathbf{V}^{(l)}, \mathbf{h}_{\text{root}}^{(l)} = f_\theta^{(\leq l)}(\mathbf{c}), \quad l = 1, \ldots, L, \tag{1}$$

yielding the cache $\mathcal{C} = \{(\mathbf{K}^{(l)}, \mathbf{V}^{(l)})\}_{l=1}^{L}$ and root hidden states $\mathbf{H}_{\text{root}} = \{\mathbf{h}_{\text{root}}^{(l)}\}$. To avoid redundant computation during the B subsequent branch comparisons, root hidden states are pre-normalised to unit vectors once and stored in this normalised form.

Exponentially weighted similarity gating. After each branch b processes its suffix tokens, ASKS computes a weighted cosine similarity between the branch's hidden states and the stored root representations. The weighting schedule places exponentially increasing emphasis on later layers, which carry more task-specific information and are therefore more discriminative for detecting genuine semantic divergence:

$$\sigma_b = \sum_{l=1}^{L} w_l \cdot \frac{\mathbf{h}_b^{(l)} \cdot \mathbf{h}_{\text{root}}^{(l)}}{\|\mathbf{h}_b^{(l)}\| \, \|\mathbf{h}_{\text{root}}^{(l)}\|}, \tag{2}$$

where $w_l = \exp(\alpha l/L) / \sum_{l'} \exp(\alpha l'/L)$ with α=1.5. We select α=1.5 by grid search over $\alpha \in \{0.5, 1.0, 1.5, 2.0\}$ on 30 held-out GPQA Diamond problems; larger values over-discount early layers and reduce sensitivity to prefix divergence, while smaller values under-weight task-specific later layers.

Branches with $\sigma_b \geq \tau$ are deemed semantically consistent with the prefix and receive the shared cache; branches below τ fall back to independent recomputation. This design strictly generalises token-exact prefix caching: any lexically identical prefix trivially achieves $\sigma_b = 1$, so every branch that vLLM or SGLang would reuse is also reused by ASKS. The converse does not hold: ASKS additionally reuses the cache for branches whose token sequences differ but whose hidden-state representations remain close to the root. In our diverse-phrasing stress test (§4.1), this generalisation yields 28.6% additional reuse beyond token-exact matching.

KV broadcast and batched decode. For branches that pass the similarity gate, the shared cache is expanded to batch size B via a zero-copy `repeat_interleave` on each layer's key and value tensors: $\mathcal{C}_B = \texttt{repeat_interleave}(\mathcal{C}, B, \text{dim}{=}0)$. An explicit `.clone()` call ensures contiguous memory layout, preventing potential SDPA kernel failures with non-contiguous inputs. All B branch suffixes are then stacked into a single batched tensor and decoded in one forward pass conditioned on $\mathcal{C}_B$: $\mathbf{y}_b = f_\theta(\mathbf{s}_b \mid \mathcal{C}_B)$, $b = 1, \ldots, B$, eliminating the $O(n^2)$ attention cost for $B{-}1$ branches.

KV probe. KV prefix sharing is not unconditionally beneficial: at short prefixes, the cost of the prefix forward pass plus the `repeat_interleave` expansion can exceed the savings. Rather than relying on the analytical threshold n^* from Proposition A.1 (which requires accurate per-platform estimates of the cost constants), RKSC includes a self-calibrating runtime probe. At the first solve call within each 64-token prefix-length bucket $k = \lfloor n/64 \rfloor$, the probe times three paths using the minimum of 3 repetitions: (i) batched full recompute, (ii) a single prefix forward, and (iii) batched suffix decode with the shared cache. KV sharing is declared beneficial iff the sum of paths (ii) and (iii) is less than path (i). On A100 with SDPA attention, the probe universally declares KV sharing beneficial at $n \geq 512$ for all five evaluated models.

2.3. CGEE: Confidence-Gated Early Exit

CGEE operates at two levels. The first level decides whether to run the verification pass at all; the second level decides how deeply to run it when it proceeds.

Generation confidence (input to Level 1). During the stepwise decode loop, RKSC records the top-1 softmax probability at each decode step for each branch. The generation confidence of branch b is the mean top-1 probability across all t steps:

$$p^{(b)} = \frac{1}{t} \sum_{j=1}^{t} \max_v \Pr[y_j^{(b)} = v], \tag{3}$$

accumulated in-place at zero additional compute cost, since the softmax is already computed to select the next token at every step. A branch that emits each token with high probability is a strong candidate for the correct answer; branches with uniformly low top-1 probabilities indicate uncertainty.

Verify-skip gate (Level 1). After decoding completes, RKSC evaluates whether to skip the verification pass entirely. The gate fires when two conditions are jointly satisfied: the winning branch has high absolute confidence, and it dominates all competing branches by a clear relative margin:

$$\max_b p^{(b)} \geq \tau_{\text{conf}} \quad \text{and} \quad \frac{\max_b p^{(b)} - \text{2nd-max}_b \, p^{(b)}}{\max_b p^{(b)}} \geq r_{\text{gap}}. \tag{4}$$

The dual condition prevents two failure modes: (a) a single branch with moderately high confidence but a close competitor (the absolute threshold alone would fire), and (b) a large relative gap between two branches that both have low absolute confidence (the gap condition alone would fire). When the gate fires, the entire δ-cost verify pass is skipped and branches are ranked directly by $p^{(b)}$.

Layer-level entropy exit (Level 2). When the verify-skip gate does not fire and the full verification forward pass begins, CGEE installs lightweight forward hooks on each transformer layer via `register_forward_hook`. At each layer l, the hook extracts the last-token hidden state, projects it through the cached unembedding matrix W_u (stored once at model load time), and computes the logit-space entropy:

$$H^{(l)} = -\sum_v \text{softmax}(\mathbf{h}^{(l)} W_u^\top)_v \, \log \text{softmax}(\mathbf{h}^{(l)} W_u^\top)_v. \tag{5}$$

The verification pass exits early at layer l^* when three conditions are jointly satisfied: (i) $l^* \geq l_{\min}$ (minimum exit depth, default 2, ensuring sufficient representation refinement); (ii) $H^{(l^*)} < \theta$ (entropy below a model-specific threshold, indicating a concentrated logit distribution); and (iii) $|H^{(l^*)} - H^{(l^*-1)}| < \epsilon$ (entropy has stabilised between consecutive layers, ruling out transient dips). When the exit triggers, the hook raises an internal exception that the solver catches, using the partial output as the verification result. Hooks are removed after each solve call to prevent memory leaks.

On Qwen2.5-7B (28 layers), the layer-level exit fires on 100% of verify passes at a mean exit layer of 18.4, confirming that the model reaches a stable logit distribution approximately 34% before the final layer. This validates the core premise of Level 2: the unembedding projection at intermediate layers contains sufficient information to determine the verification outcome, and the remaining 34% of layers are computationally wasteful for this purpose.

Calibration. The thresholds τ_{conf}, r_{gap}, and θ are calibrated per model on 30 held-out GPQA Diamond problems (non-overlapping with evaluation). τ_{conf} is set to the 75th percentile of the observed $\max_b p^{(b)}$ distribution; r_{gap} to the 25th percentile of the observed relative gap; and θ to the median of per-layer entropy values at the first stabilisation point. This yields conservative gates that fire only on clear-cut cases.

2.4. RSBCM: Reasoning-Selective Block Cache Manager

To prevent the shared KV cache from growing unboundedly under deep tree search, RKSC includes **RSBCM**. Each cached block receives an importance score $\omega = \text{branch score}/(\text{depth} + 1)$, which prioritises blocks from high-scoring branches at shallower depths. When the number of allocated blocks exceeds a configurable capacity (default 2,000 blocks), the manager evicts blocks in ascending ω order. The depth denominator ensures that blocks deep in the search tree, which are less likely to be revisited, are evicted first, while the branch-score numerator preserves blocks associated with higher-confidence reasoning trajectories.

In a stress test with max_blocks$=4 < B=8$ (chosen to force evictions within a single solve call), RSBCM fires exactly $B - 4 = 4$ evictions per problem (80 total across 20 problems, matching the expected count exactly). The overhead is $+1.5$ ms/problem (95% CI $[-6, +9]$ ms); the wide interval with $N=20$ means the overhead is consistent with zero but not provably so, the correct interpretation is that it is small relative to the speedups achieved. Answer agreement is 100% across all 20 problems. RSBCM is dormant in all single-depth experiments in this paper (total block count never exceeds the default threshold of 2,000) and is provided as a mechanism for multi-depth tree search. Full validation is in Appendix H.

Full implementation details (DynamicCache, decode loop, TF32 configuration, memory release) are in Appendix I.

3. Experimentation and Results

3.1. Experimental Setup

Hardware and software. All experiments run on a single NVIDIA A100-80GB SXM4 GPU with bfloat16 precision, TF32 matrix multiplication enabled, and the SDPA attention backend. The software stack consists of PyTorch 2.3.1, Transformers 4.44.0, and CUDA 12.1. Random seeds are fixed across all experiments (`torch.manual_seed(42)`, `numpy.random.seed(42)`, `torch.cuda.manual_seed_all(42)`).

Models. We evaluate five publicly available GQA models spanning 7B–10B parameters and three distinct GQA compression ratios: **Qwen2.5-7B-Instruct** (Team, 2024a; Yang et al., 2024) (28 layers, 28 query heads, 4 KV heads; 7:1 ratio), **Mistral-7B-Instruct-v0.3** (Jiang et al., 2023) (32L, 32Q, 8KV; 4:1), **Falcon3-7B-Instruct** (Team, 2024b) (28L, 12Q, 4KV; 3:1), **Falcon3-10B-Instruct** (Team, 2024b) (40L, 12Q, 4KV; 3:1), and **Llama-3-8B-Instruct** (AI@Meta, 2024) (32L, 32Q, 8KV; 4:1). This selection covers three training lineages (Qwen, Mistral/Meta, TII), two vocabulary sizes (32K and 128K+), and a range of head configurations, ensuring that observed gains are not artifacts of a single architecture.

Baselines. We compare against two baselines. The *No-KV* baseline batches all B branches in a single forward pass but recomputes the shared prefix independently for each branch, representing a well-engineered system without prefix caching. The *vLLM-equivalent* baseline implements token-exact prefix matching: it reuses the shared KV cache only for branches whose token sequences are byte-identical to the root prefix, mirroring the PagedAttention prefix-cache path in vLLM (Kwon et al., 2023) and SGLang (Zheng et al., 2024). In our identical-prefix experimental regime (where all branches share the same padded prefix), every branch is byte-identical to the root, so the vLLM-equivalent reuses the cache unconditionally, the same behaviour as setting $\tau=0$ in the ASKS gate. Outside this regime (the diverse-phrasing ablation in §4.1), vLLM-equivalent achieves 0% reuse by construction. Both baselines are measured through the same solver code path as RKSC to ensure timing fairness.

Calibration and evaluation sets. The 30 GPQA Diamond problems used for per-model calibration (thresholds τ_{conf}, r_{gap}, θ) are drawn from a held-out split disjoint from all evaluation sets, including the 30 problems in Table 1.

Branching and padding. All experiments use $B{=}8$ branches with eight fixed reasoning-hint suffixes cycled across branches. Problems are padded to approximately 1,024 tokens using a neutral filler string prepended iteratively, placing all experiments in the KV-beneficial regime ($n/n^* \approx 11$, Proposition A.1).

Datasets. We evaluate on four benchmarks spanning a difficulty gradient: **GPQA Diamond** (Rein et al., 2023) (graduate-level science), **MMLU-STEM** (Hendrycks et al., 2021b;a) (university-level STEM), **ARC-Challenge** (Clark et al., 2018) (medium science multiple-choice), and **GSM8K** (Cobbe et al., 2021) (elementary mathematics). Table 1 uses $N{=}30$ GPQA problems with $t{=}32$ decode steps; the extended evaluation (Table 2) uses $N{=}50$ per dataset with $t{=}8$.

Timing protocol. Each condition is timed over $N_{\text{runs}} = 2$–3 repeats per problem; the reported latency is the mean across repeats and problems. Two warmup solves per condition are executed and discarded to stabilise the CUDA allocator before timing begins, reducing the coefficient of variation from $\sim$30% to $\sim$10%. `torch.cuda.synchronize()` is called before every wall-clock measurement.

Compute budget. All experiments together consume approximately 42 A100-GPU-hours: 3 hours for calibration (5 models $\times$ 30 held-out problems $\times$ 3 conditions), 24 hours for the extended evaluation (Table 2), 9 hours for ablations (§4), and 6 hours for the throughput decomposition at $t{=}32$ (Table 1) and the verify-isolation study. Model weights are cached locally; no training occurs.

3.2. Throughput Decomposition

Table 1 isolates the contribution of each RKSC component on GPQA Diamond using Qwen2.5-7B-Instruct with $t{=}32$ decode steps. This longer decode regime makes the decode term $Bt\gamma$ non-negligible relative to the prefill term, providing a conservative estimate of gains.

Table 1. Latency decomposition on GPQA Diamond ($N{=}30$, $B{=}8$, prefix $\approx$1,024 tok, $t{=}32$).

Condition	Latency (ms)	Speedup	% Saved
No-KV *(ref)*	$1{,}956 \pm 10$	$1.000\times$	,
KV only	$1{,}457 \pm 8$	$1.343\times$	25.5%
RKSC (KV+CGEE)	$1{,}206 \pm 339$	$\mathbf{1.622}\times$	**38.3%**

KV prefix sharing alone saves 500 ms per problem (25.5%) with a bootstrap 95% CI of $[1.339, 1.347]$ and a coefficient of variation of 1%, confirming high measurement reliability. This saving is almost entirely attributable to prefix attention: at $n{=}1{,}024$, the batched prefill of $B{=}8$ branches dominates decode cost, and RKSC replaces $B{-}1$ of those prefills with a single suffix forward conditioned on the shared cache.

The combined RKSC condition (KV+CGEE) achieves $1.622\times$ ($[1.486, 1.803]$), saving 750 ms per problem. The elevated variance (CV=28%) is expected and reflects the binary nature of the CGEE skip decision, which produces a bimodal latency distribution: problems on which the gate fires land on the fast skip-path, while the rest match the KV-only condition. This bimodality inflates the standard deviation relative to the mean and does not indicate measurement instability, the KV-sharing component, which accounts for the majority of the speedup, has CV=1%. At $t{=}32$ decode steps on GPQA Diamond (the hardest dataset), the verify-skip gate fires infrequently, so CGEE's marginal contribution is modest in this decomposition. The full CGEE contribution is visible in the extended evaluation at $t{=}8$ (§3.3), where verify cost constitutes a larger fraction of total latency.

Operating-regime disclosure. The extended evaluation uses $t{=}8$ decode steps, which is representative of the first branching phase in tree search and is a favourable operating point for RKSC because a short decode keeps the verify-pass cost proportionally large. The throughput decomposition at $t{=}32$ (Table 1) and the verify-only isolation below provide complementary views at longer decode horizons where CGEE's contribution is naturally smaller.

CGEE verify-only isolation. To measure the layer-level entropy exit in isolation, we time the verification pass alone in a decode-dominated regime ($t{=}128$, $N{=}15$). Full-depth verification takes 315.3 ± 8.2 ms; CGEE-accelerated verification takes 250.7 ± 11.8 ms, yielding a verify-pass speedup of $\mathbf{1.258}\times$ (bootstrap 95% CI $[1.225, 1.293]$, $p{<}0.0001$). The layer-level entropy exit fires on 100% of branches at a mean exit layer of 18.4 out of 28, confirming that the model reaches a stable logit distribution well before the final layer on verification inputs. This validates the core premise of Level 2: the unembedding projection at intermediate layers contains sufficient information to determine the verification outcome.

3.3. Multi-Model, Multi-Dataset Evaluation

Table 2 reports the full extended evaluation. At $t{=}8$ decode steps (representative of the first branching phase in tree search, where decode cost is small relative to prefill), RKSC achieves a mean speedup of $\mathbf{3.008}\times$ across all 20 model–dataset pairs, with a peak of $\mathbf{3.990}\times$ (Llama-3-8B on MMLU-STEM).

Table 2. Extended evaluation (B=8, prefix ≈1,024 tok, t=8, N=50 per dataset, 1,000 total problems). **vLLM↑**: vLLM-equiv vs No-KV. **RKSC↑**: KV+CGEE vs No-KV. *CGEE skip*: verify-skip gate fire rate.

Model	Dataset	NoKV (ms)	vLLM (ms)	RKSC (ms)	vLLM↑	RKSC↑	*CGEE skip*
Qwen2.5-7B Instruct	GPQA	1,159	653	598	1.78×	1.94×	*3%*
	GSM8K	1,163	652	515	1.78×	2.26×	*32%*
	ARC-C	1,163	651	391	1.79×	2.97×	*71%*
	MMLU	1,160	649	371	1.79×	3.13×	*70%*
Mistral-7B Instruct	GPQA	1,257	690	590	1.82×	2.13×	*23%*
	GSM8K	1,262	689	581	1.83×	2.17×	*30%*
	ARC-C	1,263	691	482	1.83×	2.62×	*60%*
	MMLU	1,260	692	534	1.82×	2.36×	*47%*
Falcon3-7B Instruct	GPQA	1,158	637	360	1.82×	3.21×	*54%*
	GSM8K	1,170	650	376	1.80×	3.11×	*50%*
	ARC-C	1,173	651	347	1.80×	3.38×	*66%*
	MMLU	1,175	651	395	1.80×	2.98×	*39%*
Falcon3-10B Instruct	GPQA	1,629	895	450	1.82×	3.63×	*72%*
	GSM8K	1,634	898	473	1.82×	3.46×	*57%*
	ARC-C	1,628	897	418	1.82×	3.89×	*77%*
	MMLU	1,634	898	467	1.82×	3.50×	*66%*
Llama-3-8B Instruct	GPQA	1,280	708	509	1.81×	2.51×	*57%*
	GSM8K	1,283	710	464	1.81×	2.76×	*65%*
	ARC-C	1,282	712	401	1.80×	3.20×	*74%*
	MMLU	1,284	710	322	1.81×	3.99×	*92%*
Overall		**1,301**	**719**	**452**	**1.808×**	**3.008×**	*55%*

Architecture generalization. The vLLM-equivalent gain spans 1.78×–1.83× across all 20 model–dataset pairs (mean 1.808×, std 0.02), confirming the prediction of Eq. (8): since S_{KV} depends on n and B but not on model family or task, fixing the prefix length and branching factor produces near-identical gains regardless of architecture. This near-constant behaviour persists despite substantial architectural variation, five models spanning three GQA compression ratios (3:1, 4:1, 7:1), two vocabulary sizes, and three independent training lineages. The tight clustering (±0.02×) confirms that RKSC's KV throughput benefit transfers uniformly across model families without requiring a high-compression GQA architecture.

At the n=1,024 operating point, all branches achieve $\sigma_b \geq 0.95$ for all models, yielding 100% KV reuse. In this identical-prefix regime, ASKS and the vLLM-equivalent produce identical KV gains by construction; the genuine ASKS novelty over token-exact caching is isolated in §4.1.

Effect of task difficulty on CGEE. CGEE multiplies the KV-only speedup from 1.81× to 3.008× on average, a **1.66**× marginal improvement attributable entirely to the dual-level exit mechanism. The verify-skip rate $\hat{\rho}$ tracks difficulty monotonically across the four datasets: lowest on GPQA Diamond (mean ∼42% across models, where graduate-level reasoning produces high inter-branch uncertainty), and highest on ARC-Challenge and MMLU-STEM (60–92%, where shorter problems allow the model to concentrate confidence rapidly). This difficulty-adaptive behaviour follows directly from the dual condition in Eq. (4): harder problems produce smaller inter-branch gaps and lower absolute confidence, so both conditions are less likely to be satisfied simultaneously.

Effect of model scale. Falcon3-10B achieves the highest mean RKSC speedup (3.619×), compared to 2.574×–3.171× for the 7B–8B models. The absolute latency saved also scales: 1,179 ms per problem for Falcon3-10B versus 692–799 ms for smaller models. This scaling follows from the latency model (Eq. (6)): larger models have higher per-layer costs, so a fixed number of layers avoided by prefix sharing and early exit represents a proportionally larger saving. Llama-3-8B achieves the highest single-cell result (**3.990**× on MMLU-STEM with a 92% skip rate), demonstrating that models excluded from calibration benefit substantially from RKSC.

RKSC vs vLLM-equivalent. The marginal speedup of RKSC over vLLM-equivalent prefix caching averages +61.2% across all 20 model–dataset pairs. In the identical-prefix regime, this gain is attributable entirely to CGEE's dual-level exit, since ASKS and vLLM-equivalent yield identical KV gains when all branches share the same token prefix. The genuine ASKS novelty is isolated in §4.1.

3.4. Accuracy Analysis

Table 3. CGEE accuracy impact across all 20 model–dataset pairs (1,616 total verify calls).

Metric	Value
Total verify calls	1,616
CGEE-induced errors	6
Error rate	0.37%
Mean Δ accuracy	-0.2%
Mean verify-skip rate	55%

Across 1,616 verify calls spanning all five models and four datasets, CGEE introduced only 6 errors (0.37%), with a mean accuracy delta of -0.2% relative to full-depth verification (Table 3). Three model–dataset pairs account for all six errors: Mistral-7B on ARC-Challenge (3 errors, -4.0% Δaccuracy), Falcon3-10B on ARC-Challenge (2 errors, -2.0%), and Llama-3-8B on GSM8K (1 error, -2.0%). The remaining 17 model–dataset pairs exhibit zero CGEE-induced errors.

The error concentration in multiple-choice benchmarks (ARC-C) is consistent with the structure of the CGEE gate: on multiple-choice items, generation confidence reflects the model's fluency in producing the answer letter, which occasionally diverges from the correctness signal that full-depth verification captures. On free-form generation tasks (GPQA, GSM8K), where generation confidence more directly reflects understanding, the error rate is lower. These results confirm that the dual condition in Eq. (4), requiring both high absolute confidence and a clear relative margin, is an effective safeguard against false positives.

4. Analysis and Ablations

4.1. ASKS Novelty Over vLLM Prefix Caching

The experiments in §3.3 use an identical-prefix regime where both ASKS and vLLM-equivalent caching achieve 100% KV reuse and identical speedups. To isolate the contribution of semantic similarity gating, we construct a diverse-phrasing regime where branch-level token sequences differ from the first token.

We design eight phrasing templates at four semantic-divergence levels (identical, minor rewording, structural reframing, and very different academic register) and evaluate N=15 problems on Qwen2.5-7B. Token-exact caching achieves 0% reuse on all non-identical phrasings by construction, since branches diverge at token position 1. ASKS achieves 28.6% additional reuse at τ=0.82, correctly identifying branches whose hidden-state representations remain semantically consistent despite surface-level lexical divergence.

The wall-clock implication is significant: on padded diverse phrasings ($n \approx 512$), ASKS achieves a $1.131\times$ speedup over the No-KV baseline (bootstrap 95% CI $[1.066, 1.174]$, p=0.0007), while token-exact caching provides no benefit. This result demonstrates that ASKS offers a strict superset of the functionality provided by existing prefix-caching systems, with measurable throughput gains in the regime where those systems fail.

4.2. Prefix-Length Sensitivity

To characterise the operating envelope of KV sharing, we sweep the prefix length over $n \in \{128, 256, 512, 1024\}$ tokens with B=8 and N=20 problems (Qwen2.5-7B). At n=128, the speedup is $0.993\times$ ($[0.909, 1.103]$), confirming no net benefit at short prefixes where the `repeat_interleave` overhead approaches the saved prefill cost. At n=256 the result is similarly break-even ($0.999\times$, $[0.951, 1.061]$). At n=512, KV sharing is clearly beneficial ($1.127\times$, $[1.081, 1.224]$), and at n=1,024 the speedup reaches $1.452\times$ ($[1.301, 1.681]$).

The empirical transition from break-even to beneficial between n=256 and n=512 is higher than the analytical threshold $n^* \approx 90$ from Proposition A.1. This gap reflects constant-factor SDPA kernel overhead that the linear latency model does not capture: the actual prefix forward pass includes fixed-cost setup steps (kernel launch, memory allocation) that are non-negligible at short prefix lengths but amortise at $n \geq 512$. The KV probe (§2.2) handles this discrepancy correctly by measuring both paths at runtime rather than relying on the analytical threshold.

4.3. ASKS Threshold Insensitivity

In the identical-prefix regime, all branches achieve $\sigma_b \geq 0.95$ for all five models, so the ASKS gate is non-binding regardless of threshold τ. To confirm this, we sweep $\tau \in \{0.70, 0.82, 0.90, 0.95\}$ at $n \approx 512$ with N=20 problems: all four settings produce speedups of $\sim 1.16\times$ ($p < 10^{-25}$), confirming that the gate decision is τ-insensitive when the prefix is long relative to the branch suffix. This robustness is practically important: practitioners need not tune τ for their specific workload when operating at $n \geq 512$.

4.4. CGEE Skip Rate vs Task Difficulty

The CGEE verify-skip rate tracks task difficulty monotonically across the four benchmarks (averaged across all five models): GPQA Diamond 42%, GSM8K 47%, MMLU-STEM 63%, ARC-Challenge 70%. This ordering matches the expected difficulty gradient and follows directly from the gate design: on hard problems, the model distributes probability mass more uniformly across branches, causing both conditions in Eq. (4) to fail more often. On easy problems, one branch dominates early in the decode sequence, both

conditions are satisfied, and the gate fires. CGEE therefore provides its largest speedup contribution precisely on the tasks where verification is least necessary, a self-correcting property that limits accuracy risk.

5. Limitations

Sample size. Our evaluation uses $N{=}50$ problems per dataset, producing wide 95% confidence intervals on CGEE skip rates ($\pm$10–18%). A sample of $N \geq 200$ would reduce the CI to $\pm 5\%$. Importantly, the KV sharing gains (which account for the majority of the speedup) have CV$<2\%$ and are not affected by this limitation.

Model scale. We evaluate 7B–10B models on a single A100 GPU. Models at 14B+ and multi-GPU tensor-parallel configurations introduce communication overhead that may alter the relative contribution of prefix sharing versus verification cost. The latency model (Eq. (6)) predicts larger models should benefit *more*, and the Falcon3-10B results ($3.62\times$ mean vs 2.57–$3.17\times$ for 7B models) provide early evidence for this trend.

Domain coverage. Our benchmarks cover science and mathematics. The CGEE entropy threshold θ is calibrated on GPQA Diamond and may not transfer to domains with substantially different entropy profiles. Per-dataset θ calibration via a small held-out sweep (5–10 problems) restores gate activity, at the cost of one additional calibration step.

CGEE error rate. While low (0.37%, 6/1,616), CGEE is not error-free. All six errors occur in the verify-skip gate (Level 1); the layer-level entropy exit (Level 2) preserves the full verification outcome by construction. Applications requiring zero verification errors should disable Level 1 and retain only Level 2, accepting a smaller speedup contribution.

Scaling of ASKS broadcast. The `repeat_interleave` expansion is $O(BLd_k)$ in memory where d_k is the per-head KV dimension; for very large B (≥ 64) on long prefixes, this expansion can itself become non-negligible on memory-bound workloads. A zero-copy view-based implementation (rather than the safer `.clone()`) would remove this cost at the expense of additional care around SDPA kernel input contiguity.

Hook-based exit is architecture-coupled. CGEE's Level 2 exit depends on `register_forward_hook` semantics and the existence of a distinct unembedding matrix accessible at model-load time. Models with tied input/output embeddings or non-standard layer modules may require minor adapter code. We verified compatibility with all five evaluated models; extending to MoE or fused-layer architectures is future work.

Concurrent solves. RSBCM's eviction policy assumes a single solve-call owner of the cache. In a multi-tenant inference server, concurrent solves would require a tenant-aware importance score; this is supported by the current design but not empirically validated here.

6. Conclusion

We presented **RKSC**, a training-free inference acceleration framework for multi-branch LLM reasoning that addresses two structural redundancies in current inference pipelines. ASKS broadcasts a single shared prefix KV cache to all semantically consistent branches via exponentially weighted cosine similarity, generalising token-exact prefix caching to the semantic regime and demonstrating 28.6% additional KV reuse on lexically diverse branches. CGEE reduces verification cost through a dual-level exit: a verify-skip gate that bypasses the entire verification pass when generation confidence is decisive, and a layer-level entropy exit that terminates the verification forward pass at an intermediate transformer layer when per-layer logit entropy stabilises. Together, these mechanisms achieve a mean speedup of $\mathbf{3.008}\times$ (peak $\mathbf{3.990}\times$) across five model families and four benchmarks, with a CGEE-induced error rate of 0.37%. The KV sharing component is architecture-agnostic (std $0.02\times$ across five models), while the CGEE component is difficulty-adaptive (skip rates from 42% on graduate-level science to 92% on university STEM). RKSC requires no fine-tuning, no architecture modifications, and no custom CUDA kernels.

Impact Statement

This paper advances the efficiency of large language model inference for multi-step reasoning tasks. By reducing redundant computation in multi-branch reasoning pipelines, RKSC lowers the energy cost of inference-time scaling and makes state-of-the-art reasoning more accessible on constrained hardware. We are not aware of societal consequences specific to this work beyond those general to advancing the field of Machine Learning and Large Language Models (LLMs).

References

AI@Meta. Llama 3 model card. 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

An Yang, A. L., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P.,

Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Cai, T., Li, Y., Geng, Z., Peng, H., Lee, J. D., Chen, D., and Dao, T. Medusa: Simple llm inference acceleration framework with multiple decoding heads, 2024. URL https://arxiv.org/abs/2401.10774.

Chen, K., Tan, X., Yu, M., and Xu, H. Memshare: Memory efficient inference for large reasoning models through kv cache reuse, 2025a. URL https://arxiv.org/abs/2507.21433.

Chen, X., Xu, J., Liang, T., He, Z., Pang, J., Yu, D., Song, L., Liu, Q., Zhou, M., Zhang, Z., Wang, R., Tu, Z., Mi, H., and Yu, D. Do not think that much for 2+3=? on the overthinking of o1-like llms, 2025b. URL https://arxiv.org/abs/2412.21187.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Elhoushi, M., Shrivastava, A., Liskovich, D., Hosmer, B., Wasti, B., Lai, L., Mahmoud, A., Acun, B., Agarwal, S., Roman, A., Aly, A., Chen, B., and Wu, C.-J. Layerskip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12622–12642. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.681. URL http://dx.doi.org/10.18653/v1/2024.acl-long.681.

Guo, D. et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081): 633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Hendrycks, D., Burns, C., Basart, S., Critch, A., Li, J., Song, D., and Steinhardt, J. Aligning ai with shared human values. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021a.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021b.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention, 2023. URL https://arxiv.org/abs/2309.06180.

Leviathan, Y., Kalman, M., and Matias, Y. Fast inference from transformers via speculative decoding, 2023. URL https://arxiv.org/abs/2211.17192.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Miao, X., Oliaro, G., Zhang, Z., Cheng, X., Wang, Z., Zhang, Z., Wong, R. Y. Y., Zhu, A., Yang, L., Shi, X., Shi, C., Chen, Z., Arfeen, D., Abhyankar, R., and Jia, Z. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS '24, pp. 932–949. ACM, April 2024. doi: 10.1145/3620666.3651335. URL http://dx.doi.org/10.1145/3620666.3651335.

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Schuster, T., Fisch, A., Gupta, J., Dehghani, M., Bahri, D., Tran, V. Q., Tay, Y., and Metzler, D. Confident adaptive language modeling, 2022. URL https://arxiv.org/abs/2207.07061.

Team, Q. Qwen2.5: A party of foundation models, September 2024a. URL https://qwenlm.github.io/blog/qwen2.5/.

Team, T. The falcon 3 family of open models, December 2024b.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models, 2023. URL https://arxiv.org/abs/2203.11171.

Xin, J., Tang, R., Lee, J., Yu, Y., and Lin, J. Deebert: Dynamic early exiting for accelerating bert inference, 2020. URL https://arxiv.org/abs/2004.12993.

Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., Li, C., Li, C., Liu, D., Huang, F., Dong, G., Wei, H., Lin, H., Tang, J., Wang, J., Yang, J., Tu, J., Zhang, J., Ma, J., Xu, J., Zhou, J., Bai, J., He, J., Lin, J., Dang, K., Lu, K., Chen, K., Yang, K., Li, M., Xue, M., Ni, N., Zhang, P., Wang, P., Peng, R., Men, R., Gao, R., Lin, R., Wang, S., Bai, S., Tan, S., Zhu, T., Li, T., Liu, T., Ge, W., Deng, X., Zhou, X., Ren, X., Zhang, X., Wei, X., Ren, X., Fan, Y., Yao, Y., Zhang, Y., Wan, Y., Chu, Y., Liu, Y., Cui, Z., Zhang, Z., and Fan, Z. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

Yang, C., Si, Q., Duan, Y., Zhu, Z., Zhu, C., Li, Q., Chen, M., Lin, Z., and Wang, W. Dynamic early exit in reasoning models, 2025. URL https://arxiv.org/abs/2504.15895.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models, 2023. URL https://arxiv.org/abs/2305.10601.

Zheng, L., Yin, L., Xie, Z., Sun, C., Huang, J., Yu, C. H., Cao, S., Kozyrakis, C., Stoica, I., Gonzalez, J. E., Barrett, C., and Sheng, Y. Sglang: Efficient execution of structured language model programs, 2024. URL https://arxiv.org/abs/2312.07104.

The appendix is organised as follows. Appendix A contains the full theoretical analysis: the latency model, KV sharing speedup and threshold proposition, the combined RKSC speedup corollary, and the dual-level CGEE derivation. Appendix B gives detailed pseudocode for the three mechanisms referenced in §2. Appendix C collects all hyperparameters and the full prefix-length sensitivity sweep. Appendix D provides per-model accuracy breakdowns complementing Table 3. Appendix E lists the per-model cost constants used in the latency model. Appendix F shows qualitative examples of CGEE firing and deferring. Appendix G analyses per-layer entropy dynamics underlying Level-2 exit. Appendix H documents the RSBCM deep-tree validation experiment. Appendix I gives implementation details. Appendix J provides a full reproducibility checklist.

A. Theoretical Analysis

We characterise the latency savings of RKSC analytically, derive the conditions under which each mechanism is beneficial, and show that the combined speedup decomposes into factors operating on disjoint cost terms. The main-text discussion of results in §3.3 and the prefix-length ablation in §4.2 refer to Proposition A.1 and Eq. (8) from this appendix.

A.1. Latency Model

Consider generating B independent reasoning branches from a shared problem prefix of n tokens, each producing t decode tokens. We decompose wall-clock latency into four additive terms:

$$\mathcal{L} = \underbrace{\alpha n^2}_{\text{prefill}} + \underbrace{B\,\beta n}_{\text{branch overhead}} + \underbrace{B\,t\,\gamma}_{\text{decode}} + \underbrace{\delta}_{\text{verify}}, \tag{6}$$

where α is the per-token-squared self-attention cost (dominant for long prefixes due to the quadratic attention mechanism), β is the per-token feed-forward cost, γ is the per-step autoregressive decode cost, and δ is the full-depth verification forward-pass cost. The No-KV baseline pays the full prefill cost for each of the B branches: $\mathcal{L}_{\text{ref}} = B\alpha n^2 + Bt\gamma + \delta$. This model is additive in the four cost terms, enabling clean decomposition of savings.

A.2. KV Prefix Sharing Speedup

RKSC computes the prefix KV cache once (cost αn^2) and broadcasts it to all B branches, so each branch pays only the suffix attention cost βs for a short suffix of $s \ll n$ tokens:

$$\mathcal{L}_{\text{KV}} = \alpha n^2 + B\,\beta s + B\,t\,\gamma + \delta. \tag{7}$$

The KV speedup relative to the No-KV reference is:

$$S_{\text{KV}} = \frac{B\alpha n^2 + Bt\gamma + \delta}{\alpha n^2 + B\beta s + Bt\gamma + \delta}. \tag{8}$$

Proposition A.1 (KV sharing threshold)**.** *$S_{KV} > 1$ if and only if $(B-1)\alpha n^2 > B\beta s$, i.e., the shared prefix exceeds the critical length $n^* = \sqrt{B\beta s/(B-1)\alpha}$.*

Proof. Expanding $S_{\text{KV}} > 1$ gives $\mathcal{L}_{\text{ref}} > \mathcal{L}_{\text{KV}}$. Substituting Eqs. (7) and the No-KV reference, the $Bt\gamma$ and δ terms cancel, leaving $(B-1)\alpha n^2 > B\beta s$. Solving for n yields $n > \sqrt{B\beta s/(B-1)\alpha} = n^*$. □

Equation (8) has two important structural properties. First, S_{KV} depends on n, B, s, and the cost ratios α/β and α/γ, but *not* on model family, GQA compression ratio, or task. This predicts that fixing n and B produces near-identical KV gains regardless of architecture, a prediction verified empirically in §3.3, where the vLLM-equivalent gain varies by only $\pm 0.03\times$ across five heterogeneous model families. Second, S_{KV} increases monotonically in n and B but decreases in t: longer decode sequences dilute the KV-dominated prefill fraction. On A100 with SDPA, empirical timing on Qwen2.5-7B gives $n^* \approx 90$ tokens, placing all experiments at n=1,024 in the beneficial regime ($n/n^* \approx 11$).

Worked numerical example. The per-token self-attention cost α and per-token feed-forward cost β can be fit by microbenchmarking the prefill path at varying n and recovering the quadratic and linear coefficients. Plugging empirical values into Proposition A.1 with B=8 and s=16 yields $n^* \approx 90$ tokens once kernel-launch overhead is folded in. The

empirical break-even (§4.2) is higher (n between 256 and 512) because fixed SDPA setup costs dominate the linear model at short prefixes, precisely why RKSC uses a runtime probe (§2.2) rather than the analytical threshold alone. Fitted cost constants per model appear in Appendix E.

A.3. Dual-Level CGEE Speedup

CGEE saves latency through two independent channels. The verify-skip gate (Level 1) saves the full δ on a fraction ρ of problems. The layer-level entropy exit (Level 2) saves a fraction $(1 - l^*/L)$ of δ on a fraction ϕ of the remaining $(1 - \rho)$ problems where entropy stabilises before the final layer. The combined expected latency with both CGEE levels active is:

$$\mathbb{E}[\mathcal{L}_{\text{CGEE}}] = \mathcal{L}_{\text{KV}} - \rho\,\delta - (1{-}\rho)\,\phi\,\delta\,(1 - l^*/L), \tag{9}$$

and the full RKSC speedup relative to the No-KV baseline is:

$$S_{\text{RKSC}} = \frac{\mathcal{L}_{\text{ref}}}{\mathbb{E}[\mathcal{L}_{\text{CGEE}}]} = \frac{B\alpha n^2 + Bt\gamma + \delta}{\alpha n^2 + B\beta s + Bt\gamma + \delta - \rho\delta - (1{-}\rho)\phi\delta(1 - l^*/L)}. \tag{10}$$

Corollary A.2 (Combined RKSC speedup)**.** *Let ρ denote the fraction of problems on which the verify-skip gate fires, and let the layer-level entropy exit fire at layer l^* on a fraction ϕ of the remaining $(1{-}\rho)$ problems. Then S_{RKSC} is given by Eq. (10), and $S_{RKSC} \geq S_{KV}$ with equality iff $\rho = \phi = 0$.*

Proof. Substitute Eq. (9) into $S_{\text{RKSC}} = \mathcal{L}_{\text{ref}}/\mathbb{E}[\mathcal{L}_{\text{CGEE}}]$ to obtain Eq. (10). Both subtractive terms in the denominator are non-negative, so $\mathbb{E}[\mathcal{L}_{\text{CGEE}}] \leq \mathcal{L}_{\text{KV}}$, giving $S_{\text{RKSC}} \geq S_{\text{KV}}$. Equality holds iff $\rho\delta + (1{-}\rho)\phi\delta(1{-}l^*/L) = 0$, i.e. iff $\rho = \phi = 0$ (since $\delta > 0$ and $l^* < L$). □

A.4. Structural Interpretation of the Speedup Decomposition

Equation (10) decomposes the total latency savings into three mutually disjoint cost terms:

- The KV-sharing term $(B{-}1)\alpha n^2 - B\beta s$, which removes $B{-}1$ of the B prefill costs at the expense of a suffix attention cost. This term is active on *every* solve call when $\sigma_b \geq \tau$.
- The verify-skip term $\rho\delta$, which removes the entire verify pass on a ρ-fraction of problems. This term is active only on problems where generation confidence is decisive.
- The layer-exit term $(1{-}\rho)\phi\delta(1 - l^*/L)$, which removes a $(1 - l^*/L)$-fraction of the verify pass on the remaining problems. This term is active whenever verify-skip does not fire but layer-level entropy stabilises.

Because the three savings act on disjoint cost terms (αn^2, δ in full, and δ in part), they compose additively in the speedup ratio rather than trading off. This is the structural reason that combining the three mechanisms produces the observed $1.66\times$ multiplicative improvement over vLLM-equivalent caching.

A.5. Skip-Rate Adaptivity

The skip rate ρ is an emergent property of the model–task interaction. On hard problems, $\max_b p^{(b)}$ is low, inter-branch gaps are small, and ρ is correspondingly low. On easier tasks, the model rapidly concentrates confidence on one branch, producing large gaps and high ρ. This difficulty-adaptive behaviour means CGEE applies aggressive savings when the model is most confident, and defers to full verification when uncertainty warrants it, matching the qualitative ordering of task difficulty reported in §4.4.

B. Algorithmic Details

This appendix provides detailed pseudocode for the three RKSC mechanisms. Algorithm 2 details the ASKS similarity gate; Algorithm 3 details the CGEE layer-level entropy exit; Algorithm 4 details RSBCM eviction.

Algorithm 2 ASKSGATE: similarity-gated KV reuse

Require: root hidden states $\mathbf{H}_{\text{root}} = \{\mathbf{h}_{\text{root}}^{(l)}\}_{l=1}^{L}$ (pre-normalised), branch suffixes $\{\mathbf{s}_b\}_{b=1}^{B}$, threshold τ, exponent α_w

compute weights $w_l \leftarrow \exp(\alpha_w l/L)/\sum_{l'=1}^{L}\exp(\alpha_w l'/L)$ for $l = 1, \ldots, L$
sharedSet $\leftarrow \emptyset$
for $b = 1, \ldots, B$ **do**
  $\{\mathbf{h}_b^{(l)}\} \leftarrow f_\theta^{\text{suffix}}(\mathbf{s}_b)$ {per-layer hidden states for this suffix}
  $\sigma_b \leftarrow \sum_{l=1}^{L} w_l \cdot \langle \mathbf{h}_b^{(l)}/\|\mathbf{h}_b^{(l)}\|, \mathbf{h}_{\text{root}}^{(l)}\rangle$
  if $\sigma_b \geq \tau$ **then**
    sharedSet.add(b)
  end if
end for
return sharedSet

Algorithm 3 VERIFYWITHENTROPYEXIT: layer-level entropy-stabilisation exit

Require: candidate tokens $\{\mathbf{y}_b\}$, unembedding W_u, thresholds θ, ϵ, $l_{\min}$

install forward hooks on each transformer layer (via `register_forward_hook`)
$H_{\text{prev}} \leftarrow +\infty$
for $l = 1, \ldots, L$ **do**
  compute layer output $\mathbf{h}^{(l)}$ (forward pass up to layer l)
  $\mathbf{z} \leftarrow \mathbf{h}_{\text{last-token}}^{(l)} \cdot W_u^\top$ {unembedding projection}
  $\mathbf{p} \leftarrow \mathrm{softmax}(\mathbf{z})$
  $H^{(l)} \leftarrow -\sum_v \mathbf{p}_v \log \mathbf{p}_v$
  if $l \geq l_{\min}$ **and** $H^{(l)} < \theta$ **and** $|H^{(l)} - H_{\text{prev}}| < \epsilon$ **then**
    raise `_EarlyExitSignal`($\mathbf{p}$) {caught by solver; hooks removed in finally block}
  end if
  $H_{\text{prev}} \leftarrow H^{(l)}$
end for
remove hooks; **return** $\mathrm{softmax}(\mathbf{h}^{(L)} W_u^\top)$ {full-depth result if no early exit}

C. Hyperparameter Settings and Sensitivity

C.1. Complete Hyperparameter Table

Table 4 collects every hyperparameter used in RKSC, split into fixed-across-models settings (top block) and per-model calibrated settings (bottom block). Calibrated thresholds are obtained on 30 held-out GPQA Diamond problems disjoint from the evaluation set.

The τ_{conf} and r_{gap} values are shared across all models because they are set to percentiles of each model's own confidence distribution, making them self-normalising. The θ values differ because Mistral-7B's logit distributions are substantially sharper (lower entropy) than the other models.

C.2. Prefix-Length Sweep

C.3. ASKS Threshold Sweep

Table 6 reports the full τ-sweep referenced in §4.3. In the identical-prefix regime, all four settings produce statistically indistinguishable speedups at $n \approx 512$, confirming that the gate decision is τ-insensitive in this regime.

C.4. CGEE Cross-Dataset Calibration

The entropy threshold θ is calibrated on GPQA Diamond. Table 7 shows the consequences of using a single GPQA-calibrated θ versus per-dataset calibration. Per-dataset θ calibration via a small held-out sweep (5–10 problems) restores gate activity on out-of-distribution datasets without loss in accuracy.

Algorithm 4 RSBCMEVICT: attention-weighted depth-priority eviction

Require: block table $\mathcal{B} = \{(\text{block}_i, \text{score}_i, \text{depth}_i)\}$, capacity $N_{\max}$
while $|\mathcal{B}| > N_{\max}$ **do**
  $\omega_i \leftarrow \text{score}_i/(\text{depth}_i + 1)$ for all i
  $i^* \leftarrow \arg\min_i \omega_i$
  evict block i^* from cache
  $\mathcal{B} \leftarrow \mathcal{B} \setminus \{(\text{block}_{i^*}, \text{score}_{i^*}, \text{depth}_{i^*})\}$
end while

This cross-domain behaviour is documented as a limitation in §5. The main-paper results use per-dataset θ throughout; the GPQA-calibrated value is the *default* for deployment without a calibration sweep.

D. Per-Model Accuracy Analysis

Table 8 decomposes the aggregate accuracy impact (Table 3) across the 20 model–dataset pairs. Six errors concentrate in three pairs; the remaining 17 pairs exhibit zero CGEE-induced errors relative to full-depth verification.

D.1. Failure Case Analysis

Examining the six CGEE-induced errors reveals a consistent pattern: in each case, one branch emits the answer letter with high absolute probability while simultaneously carrying a subtle formatting deviation (e.g., a trailing period, different case, or embedded justification) that full-depth verification penalises but generation confidence does not. On ARC-Challenge, this manifests as the model fluently emitting, say, "C." with $p > 0.9$ while full-depth verification prefers the branch that emitted "C" without punctuation because the verifier has learned the benchmark's exact-match convention.

This is not a failure of the entropy signal itself but of the verify-skip gate's reliance on generation confidence as a *proxy* for correctness on multiple-choice tasks. On free-form tasks (GSM8K, GPQA), where the answer letter is embedded in longer reasoning and generation confidence more directly reflects understanding, the error rate drops to $0{-}1$ per 50 problems. The dual condition in Eq. (4), requiring both high absolute confidence and a clear relative margin, prevents the naive failure mode of a single low-confidence branch firing the skip, but does not guard against correlated fluency-vs-correctness divergence on format-sensitive tasks. Users running on such benchmarks should either raise τ_{conf} or disable Level 1 and rely on Level 2 alone.

E. Empirical Cost Constants

Table 9 reports the KV probe measurements for all five models, obtained by running the self-calibrating runtime probe (§2.2) at $n{\approx}1{,}024$ tokens, $B{=}8$. The probe times three paths, batched full recompute, single root forward, and batched suffix decode with shared cache, and confirms that KV sharing is beneficial for all models at this prefix length.

All five models achieve $B^* \leq 1.7$, confirming KV sharing is beneficial for any $B \geq 2$. At the experimental setting of $B{=}8$, the net saving per solve call ranges from 445 ms (Falcon3-7B) to 627 ms (Falcon3-10B), consistent with the latency model prediction that larger models benefit more (Eq. (8)). The near-identical B^* values (1.6–1.7) across architectures spanning three training lineages, two vocabulary sizes, and 7B–10B parameters confirms the architecture-invariance claim in §3.3: the break-even condition $(B{-}1)\alpha n^2 > B\beta s$ (Proposition A.1) depends on the cost ratio α/β, which is approximately constant across GQA model families at fixed hardware.

F. Qualitative Examples

We present two contrasting qualitative examples to illustrate when CGEE fires and when it defers.

Example 1: CGEE-skip fires (ARC-Challenge, Qwen2.5-7B).

> *Problem:* Which of the following is the primary greenhouse gas released by livestock digestion? (A) Carbon dioxide (B) Methane (C) Water vapour (D) Nitrous oxide

Table 4. Complete RKSC hyperparameter settings. Values marked *calibrated* are fit per model on 30 held-out problems.

Hyperparameter	Symbol	Value	Role
Fixed (shared across all models)			
Branching factor	B	8	number of parallel reasoning branches
Suffix length	s	16 tokens	per-branch reasoning hint length
Decode steps (main eval)	t	8	Table 2
Decode steps (throughput table)	t	32	Table 1
Decode steps (verify isolation)	t	128	§3.2
ASKS exponent	α_w	1.5	grid-searched on $\{0.5, 1.0, 1.5, 2.0\}$
ASKS similarity threshold	τ	0.82	validated insensitive (§4.3)
Min exit layer (CGEE)	$l_{\min}$	2	representation-refinement guard
Entropy stabilisation tolerance	ϵ	3.0 (nats)	rules out transient entropy dips
RSBCM default capacity	$N_{\max}$	2,000 blocks	overridden to 4 in RSBCM stress test
Decode-level CGEE threshold	θ_{dec}	0.72	early termination in decode loop
Decode-level CGEE gap	r_{dec}	0.10	decode-loop gap condition
KV probe bucket width	,	64 tokens	prefix-length probe granularity
KV probe repetitions	r_{probe}	3	min of 3 timings per path
Warmup solves	N_{warmup}	2	per condition
Timing repetitions	N_{runs}	2–3	per problem, per condition
Calibrated per model (75th pct of $\max p$ */ 25th pct of gap on 30 held-out GPQA problems, disjoint from all evaluation sets)*			
τ_{conf} (all models)	τ_{conf}	0.70	Level-1 absolute confidence threshold
r_{gap} (all models)	r_{gap}	0.06	Level-1 relative gap threshold
θ (Qwen2.5-7B)	θ	8.0	Level-2 entropy threshold (nats)
θ (Mistral-7B)	θ	0.50	Level-2 entropy threshold (nats)
θ (Falcon3-7B)	θ	8.0	Level-2 entropy threshold (nats)
θ (Falcon3-10B)	θ	8.0	Level-2 entropy threshold (nats)
θ (Llama-3-8B)	θ	8.0	Level-2 entropy threshold (nats)
ASKS τ (Qwen2.5-7B)	τ	0.82	ASKS similarity gate threshold
ASKS τ (Mistral-7B)	τ	0.80	ASKS similarity gate threshold
ASKS τ (Falcon3-7B)	τ	0.80	ASKS similarity gate threshold
ASKS τ (Falcon3-10B)	τ	0.80	ASKS similarity gate threshold
ASKS τ (Llama-3-8B)	τ	0.82	ASKS similarity gate threshold (default)

Table 5. End-to-end speedup vs prefix length (Qwen2.5-7B, N=20, B=8).

Prefix n	Speedup	95% CI	CV	Reuse
128	$0.993\times$	$[0.909, 1.103]$	5%	100%
256	$0.999\times$	$[0.951, 1.061]$	2%	100%
512	$1.127\times$	$[1.081, 1.224]$	1%	100%
1024	$1.452\times$	$[1.301, 1.681]$	0%	100%

All 8 branches converge on "(B) Methane" within 3 decode steps. Top-1 probabilities: $\max_b p^{(b)} = 0.94$; 2nd-max $= 0.22$; gap ratio $= 0.77$. Both conditions of Eq. (4) fire and the verify pass is skipped entirely. Full-depth verify, run separately, confirms branch 3 is the correct answer, agreeing with the skipped decision. Latency: 321 ms (skip) vs 512 ms (full verify), a $1.59\times$ reduction on this single problem.

Example 2: CGEE-skip defers (GPQA Diamond, Qwen2.5-7B).

> *Problem:* A particle of mass m is placed in a 3D isotropic harmonic oscillator potential with frequency ω. What is the degeneracy of the first excited state?

Branches split: 4 emit "3", 3 emit "6", 1 emits "2". $\max_b p^{(b)} = 0.61$; 2nd-max $= 0.54$; gap ratio $= 0.11 < r_{\text{gap}}$. The relative-gap condition fails, so the gate defers to full verification. Level 2 (layer-level entropy exit) fires at layer 19 out of 28, saving $\approx 32\%$ of the verify cost. Full verify selects the "3" branches (correct: the first excited state has degeneracy 3 for the 3D isotropic oscillator). Latency: 641 ms (Level-2 exit) vs 892 ms (full verify), a $1.39\times$ reduction. This example illustrates the difficulty-adaptive behaviour quantified in §4.4.

Table 6. ASKS threshold sweep at $n \approx 512$, N=20 problems on Qwen2.5-7B. Speedups are indistinguishable at all four settings, confirming τ-insensitivity in the identical-prefix regime.

τ	Speedup vs No-KV	95% CI	Reuse rate
0.70	1.162×	[1.155, 1.170]	100%
0.82	1.163×	[1.154, 1.171]	100%
0.90	1.163×	[1.154, 1.171]	100%
0.95	1.158×	[1.150, 1.167]	100%

Table 7. CGEE Level-2 exit activity: GPQA-calibrated θ vs per-dataset-calibrated θ (Qwen2.5-7B). The GPQA-calibrated θ=8.0 does not activate on out-of-distribution datasets with lower entropy profiles; per-dataset calibration (5–10 problems) restores full activity.

Dataset	Exit rate (GPQA-θ)	Exit rate (per-dataset-θ)	Note
GPQA Diamond (calib.)	100%	100%	calibration target
GSM8K	0%	100%	lower entropy; needs $\theta_{\text{GSM}} < 8.0$
ARC-Challenge	0%	100%	lower entropy; needs $\theta_{\text{ARC}} < 8.0$
MMLU-STEM	0%	100%	lower entropy; needs $\theta_{\text{MMLU}} < 8.0$

These examples are illustrative and the specific latencies quoted are from individual runs rather than averages; aggregate figures appear in Table 2.

G. Per-Layer Entropy Dynamics

Table 10 shows the typical per-layer entropy trajectory during a verification forward pass on Qwen2.5-7B (28 layers). The trajectory confirms the premise of Level 2: entropy decays approximately monotonically through the network, with a rapid drop in the middle layers followed by a plateau that begins around layer 17–19 and persists to the final layer.

The stabilisation condition $|H^{(l)} - H^{(l-1)}| < \epsilon = 3.0$ is first satisfied at layer $l^* = 19$ on average, matching the reported mean exit layer of 18.4 in the main text. Between layer 19 and layer 28, entropy changes by less than 0.04 nats, well below any decision-changing threshold for the argmax over the vocabulary on this task, confirming that the remaining 34% of layers do not alter the verification outcome and can safely be skipped. The values in Table 10 are representative means from profiling runs on GPQA Diamond hold-out problems; exact per-problem curves are available in the released code.

H. RSBCM Deep-Tree Validation

To validate RSBCM in the regime where evictions actually fire, we run a stress-test configuration with capacity $N_{\max} = 4$ and branching $B = 8$ on 20 GPQA Diamond problems, forcing exactly $B - N_{\max} = 4$ evictions per solve call. Table 11 reports the results.

The eviction count matches the expected value exactly, confirming that the importance score $\omega = \text{score}/(\text{depth} + 1)$ correctly identifies and retires the lowest-value blocks. Overhead is statistically indistinguishable from zero, and no answer regressions were observed. RSBCM is therefore dormant in the default single-depth experiments reported in the main paper (where $B < N_{\max} = 2000$ always holds), but available as a safety mechanism for deep tree search.

I. Implementation Details

I.1. Stepwise Batched Decode

RKSC replaces `generate()` with a direct `model()` loop: (1) suffix prefill conditioned on $\mathcal{C}_B$; (2) token-by-token decode with pre-allocated attention masks sliced per step; (3) early termination on EOS or decode-level CGEE gate (θ_{dec}=0.72, r_{dec}=0.10). Eliminating `GenerationMixin` scaffolding removes logit processors, stopping criteria, and beam tracking that are not needed in this pipeline, saving approximately $8{-}12\%$ of end-to-end decode time measured on Qwen2.5-7B.

Table 8. Per-model, per-dataset CGEE accuracy impact. **Errors**: number of problems where CGEE's answer diverges from full-depth verify. **ΔAcc**: CGEE accuracy minus full-verify accuracy (N=50 per cell).

Model	Dataset	Errors	ΔAcc	Skip rate
Qwen2.5-7B	GPQA	0	0.0%	3%
	GSM8K	0	0.0%	32%
	ARC-C	0	0.0%	71%
	MMLU	0	0.0%	70%
Mistral-7B	GPQA	0	0.0%	23%
	GSM8K	0	0.0%	30%
	ARC-C	**3**	−4.0%	60%
	MMLU	0	0.0%	47%
Falcon3-7B	GPQA	0	0.0%	54%
	GSM8K	0	0.0%	50%
	ARC-C	0	0.0%	66%
	MMLU	0	0.0%	39%
Falcon3-10B	GPQA	0	0.0%	72%
	GSM8K	0	0.0%	57%
	ARC-C	**2**	−2.0%	77%
	MMLU	0	0.0%	66%
Llama-3-8B	GPQA	0	0.0%	57%
	GSM8K	**1**	−2.0%	65%
	ARC-C	0	0.0%	74%
	MMLU	0	0.0%	92%
Total	,	**6**	**−0.2% (mean)**	**55% (mean)**

Table 9. KV probe measurements on A100-80GB with SDPA, B=8, n≈1,024 tokens. *batch_full*: batched prefill with no sharing (ms). *prefix_fwd*: single root forward pass (ms). *sfx+KV*: batched suffix decode conditioned on shared cache (ms). *net*: latency saved per solve call (ms). B^*: break-even branching factor; KV sharing is beneficial for all $B > B^*$.

Model	batch_full	prefix_fwd	sfx+KV	net saved	B^*
Qwen2.5-7B	586.8	88.1	43.1	455.7	1.6
Mistral-7B	629.4	89.9	58.8	480.7	1.7
Falcon3-7B	567.4	73.5	48.7	445.2	1.6
Falcon3-10B	798.7	103.4	68.8	626.5	1.6
Llama-3-8B	606.4	88.9	58.1	459.4	1.7
Verify isolation (Qwen2.5-7B, t=128): full 315.3±8.2 *ms; CGEE* 250.7±11.8 *ms (*1.258×*, p<0.0001)*					

I.2. CGEE Layer Hooks

Hooks are installed via `register_forward_hook` on each transformer layer module. Each hook extracts the last-token hidden state, projects it through the cached unembedding matrix W_u, computes softmax entropy, and raises an `_EarlyExitSignal` exception when exit conditions are met. The solver catches this exception in a `try/finally` block that guarantees hook removal even if an exit fires, preventing a memory leak that would otherwise persist across solve calls. On Qwen2.5-7B, the per-layer hook overhead is $\approx 40\,\mu$s, which is recovered many times over whenever an exit fires before the final layer.

I.3. KV Cache Expansion

The prefix cache is expanded via `.repeat_interleave(B, dim=0)` with `.clone()` for contiguous memory. Direct views (without `.clone()`) exposed an SDPA kernel bug on non-contiguous inputs in our PyTorch 2.3.1 build; the explicit clone resolves this at the cost of one additional tensor allocation per solve call, which is negligible at $n = 1024$ but grows with $B \cdot L \cdot d_k$.

Table 10. Typical per-layer entropy trajectory on Qwen2.5-7B verify pass (means over 30 held-out GPQA problems). *Layer* denotes the transformer layer index (1–28); *Entropy* is $H^{(l)}$ from Eq. (5).

Layer	Entropy $H^{(l)}$	$\lvert H^{(l)} - H^{(l-1)} \rvert$
1	9.82	,
4	7.41	0.81
8	5.12	0.62
12	3.08	0.45
16	1.62	0.23
18	1.14	0.07
19	**1.08**	**0.006**
20	1.06	0.002
24	1.04	0.001
28	1.04	0.000

Table 11. RSBCM stress test on Qwen2.5-7B (N=20 problems, B=8, $N_{\max}$=4).

Metric	Value
Expected evictions	$20 \times (B - N_{\max}) = 80$
Observed evictions	80
Overhead per problem	$+1.5$ ms
95% CI on overhead	$[-6, +9]$ ms
Interpretation	overhead small relative to savings; N=20 too small to bound to zero
Answer agreement vs no-eviction baseline	100% (20/20)
Wall-clock cost of 80 evictions	30 ms total

I.4. Memory Release

`unload_model()` explicitly pops `arch['_unembed_weight']` before deleting the model, preventing a $\sim$1.8 GB reference leak for 9B models, then runs 3 GC passes and `torch.cuda.empty_cache()`. Without this explicit pop, PyTorch's reference counting retains the unembedding matrix via the cached reference in the CGEE hook infrastructure, causing accumulating leaks across sequential solves.

I.5. TF32 and bfloat16 Configuration

All experiments use bfloat16 model weights with TF32-enabled matrix multiplication (`torch.backends.cuda.matmul.allow_tf32 = True`) and SDPA attention (`scaled_dot_product_attention`). TF32 provides approximately 5% additional throughput at negligible precision loss; disabling it produces identical answers but $\approx 5\%$ higher latency across all conditions uniformly (so the reported *ratios* are TF32-independent).

J. Reproducibility Checklist

Hardware and software.

- Single NVIDIA A100-80GB SXM4 GPU, PCIe 4.0.
- PyTorch 2.3.1, Transformers 4.44.0, CUDA 12.1, SDPA attention backend, bfloat16 + TF32.
- Ubuntu 22.04, Python 3.10.

Determinism.

- Seeds: `torch.manual_seed(42)`, `numpy.random.seed(42)`, `torch.cuda.manual_seed_all(42)`, `random.seed(42)` at the start of every condition.
- `torch.backends.cudnn.deterministic = True` and `torch.backends.cudnn.benchmark =`

`False` for the warmup phase; benchmark re-enabled for timing measurements to allow kernel autotuning (identical kernel across all conditions ensures timing fairness).

Timing.

- $N_{\text{runs}} = 2$–3 per problem, $N_{\text{warmup}} = 2$ per condition, `torch.cuda.synchronize()` before each measurement.
- Warmup discarded; reported latency is mean over the N_{runs} measurements and problems.
- Bootstrap 95% confidence intervals via $B_{\text{boot}} = 10{,}000$ resamples of the problem-level latencies.

Data.

- Datasets: GPQA Diamond (rein2023), GSM8K (cobbe2021), ARC-Challenge (allenai:arc), MMLU-STEM (hendrycks-stest2021) via HuggingFace Datasets.
- Evaluation sets: first N=50 problems per dataset (main eval), first N=30 for throughput decomposition, N=15 for verify isolation, N=20 for prefix-length sweep. Fixed indices used across conditions to ensure paired comparisons.
- Held-out calibration set: 30 GPQA Diamond problems disjoint from evaluation.
- Padding: neutral filler string prepended iteratively to $\approx$1,024 tokens; hard cap at max_seq_len $-$ 256 tokens to reserve decode headroom.

Models.

- All models loaded with `AutoModelForCausalLM.from_pretrained(..., torch_dtype=torch.bfloat16, attn_implementation="sdpa")`.
- Checkpoints: Qwen2.5-7B-Instruct, Mistral-7B-Instruct-v0.3, Falcon3-7B-Instruct, Falcon3-10B-Instruct, Meta-Llama-3-8B-Instruct (HuggingFace hub).
- Chat templates: each model's native `apply_chat_template` used verbatim.

Branching.

- $B = 8$ branches throughout; 8 fixed reasoning-hint suffixes cycled across branches (suffixes listed verbatim in released code).

Code and artefacts.

- Code release: https://github.com/AnirudhSekar/RKSC.
- Released artefacts include: the RKSC solver, calibration scripts, all timing harnesses, bootstrap analysis scripts, and per-problem latency logs.

Expected compute. Reproducing the full set of main-paper results requires approximately 42 A100-GPU-hours (itemised in §3.1).