KV-RESCUE: RECOVERING REASONING LANGUAGE MODEL KV EVICTION LOSS VIA STEPWISE INTERLEAVING

Anonymous authors
Paper under double-blind review

ABSTRACT

KV-cache eviction caps the memory cost of long reasoning traces but is inherently lossy because the model decodes from a partial view of its history. Under aggressive budgets, this not only lowers accuracy but can also cause runaway degeneration, where the model produces incoherent or repetitive tokens until reaching the length limit. We characterize much of this loss as an *information gap* caused by missing context, rather than a *capability gap* caused by limited model capacity. An evicted 7B model and a full-context 1.5B model make complementary errors, and an oracle choice between their answers recovers 79% of the accuracy gap to the full-KV 7B model. Based on this observation, we propose **KV-Rescue**, a training-free inference framework that bridges the information gap introduced by KV eviction using a lightweight full-context helper. KV-Rescue interleaves reasoning steps from the two models into a shared trajectory. An online detector uses entropy and compressibility to terminate the generation of incoherent or repetitive base-model candidates early. Across five math benchmarks with Qwen2.5-Math 7B and 72B, KV-Rescue recovers an average of 87% of the accuracy lost to eviction at budget $B = 64$. A decode-cost analysis further shows that preventing runaway degeneration cuts base-model token generation by 43% on average.

1 INTRODUCTION

The growing use of long-context reasoning and agentic systems makes the KV cache a major bottleneck in LLM inference. KV-cache compression methods address this bottleneck, including quantization Hooper et al. (2024); Liu et al. (2024); Son et al. (2025) and merging Wang et al. (2024b); among them, KV eviction Zhang et al. (2023); Li et al. (2024) is one of the most actively studied. Recent work tailors eviction to reasoning models by exploiting their structure Cai et al. (2025) and reports that aggressive eviction can induce degeneration: the model falls into repetition loops or, in severe cases, emits incoherent tokens until reaching the length limit. Eviction is still inherently lossy, since the model sees only a selected part of the context rather than its full history, and the problem grows more acute at low budgets.

Eviction creates an information gap distinct from capability. The accuracy lost to eviction is an *information gap*: the model cannot see parts of its past context. A small model underperforming a large one instead reflects a *capability gap*: it sees the same information but has less capacity to solve the problem. These gaps arise from different limitations, so a small full-context model and an evicted large model can make complementary errors. Figure 1(a) supports this view: an oracle choice between an evicted 7B model and a full-context 1.5B model recovers 79% of the accuracy gap to the full-KV 7B model.

KV Eviction and Test-Time Scaling. It remains unclear whether test-time scaling alone can recover the loss caused by eviction. We first apply stepwise best-of-N to the evicted model. As Figure 1(b) shows, increasing N improves accuracy but eventually plateaus below full-KV. All candidates are sampled from the same partial cache state, so additional sampling does not restore the missing context.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

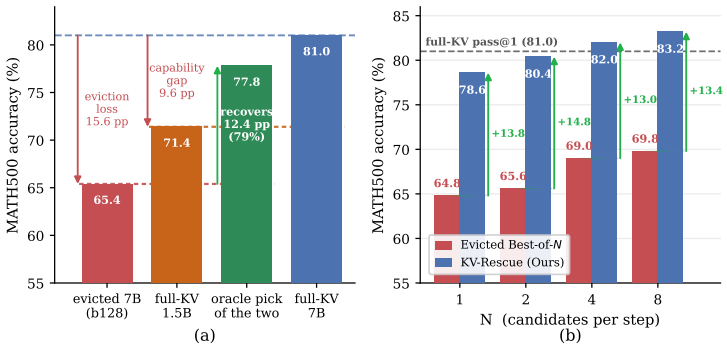


Figure 1: **(a)** An evicted 7B and a full-KV 1.5B make complementary errors; an oracle pick between them recovers 79% of the eviction loss. **(b)** Best-of-N plateaus below the single-candidate full-KV reference, while KV-Rescue (ours) recovers most of the gap and wins at every N. (MATH500, Qwen2.5-Math-7B, budget 128.)

Motivated by these observations, we propose **KV-Rescue**, a training-free inference framework that bridges the information gap introduced by KV eviction with a lightweight full-context helper. KV-Rescue interleaves reasoning steps from the evicted base model and the helper. At each step, both models generate candidates from their respective cache states, and a process reward model selects the step that extends the shared trajectory. We generalize this procedure to a best-of-N setting that draws several candidates from each model. On MATH500 at budget 128 (Figure 1(b)), KV-Rescue reaches the single-candidate full-KV reference by $N{=}4$ and continues to improve as N increases. Across five math benchmarks with Qwen2.5-Math 7B and 72B, KV-Rescue recovers an average of 87% of the accuracy lost to eviction at budget $B = 64$. It also suppresses runaway degeneration; in our decode-cost analysis, this reduces base-model token generation by 43% on average.

Our contributions are as follows:

- We identify KV eviction loss as an *information gap* distinct from a model's *capability gap*, and show that an evicted large model and a small full-context model make complementary errors.
- We introduce **KV-Rescue**, a training-free inference framework for context-complementary reasoning between an evicted base model and a lightweight full-context helper, with an online detector that terminates degenerate base candidates during generation.
- We evaluate KV-Rescue across five math benchmarks, two base-model sizes, and two eviction methods, showing that its recovery persists across model scales and eviction policies while controlling runaway degeneration.

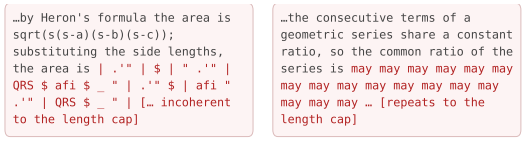
2 Preliminaries

2.1 KV Eviction

At decoding step t, a Transformer caches the key–value pairs of all preceding tokens, $\mathcal{C}_t = \{(k_i, v_i)\}_{i<t}$, where layer and attention-head indices are omitted for simplicity. Because the cache grows linearly with sequence length, it becomes a major memory cost for long reasoning traces. Abstractly, KV eviction (Cai et al., 2025; Li et al., 2024; Zhang et al., 2023) assigns each entry an importance score s_i, such as accumulated attention, and retains only the B highest-scoring entries:

$$\mathcal{C}_t^B = \{(k_i, v_i) : i \in \mathrm{TopB}(\{s_j\}_{j<t})\}\,. \tag{1}$$

The model attends to $\mathcal{C}_t^B$ instead of the full cache $\mathcal{C}_t$. Since discarded entries are unavailable without recomputation or retrieval, reducing B restricts the historical information available to subsequent predictions.



(a) incoherent generation (b) repetition loop

Figure 2: Two degeneration modes observed under aggressive KV eviction in Qwen2.5-Math-7B stepwise best-of-N decoding traces. Coherent reasoning (gray) transitions into (a) incoherent generation or (b) a repetition loop, and continues to the length limit without producing an answer.

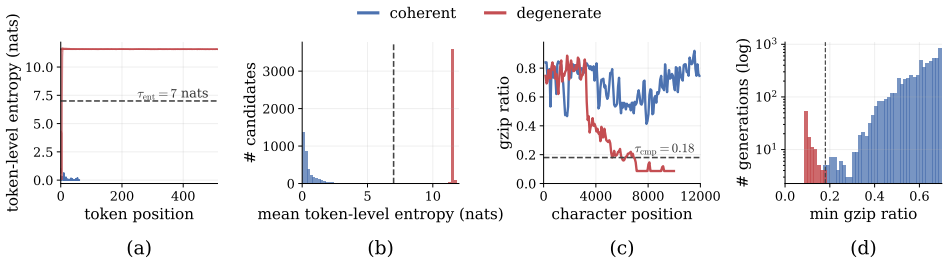


Figure 3: Online degeneration detection (Qwen2.5-Math-7B, R-KV eviction). Both signals are computed online without additional model forward passes. **(a,b)** Incoherent generation has sustained high token-level entropy, and τ_{ent} lies in the gap between coherent and incoherent candidates. **(c,d)** Repetition loops are confident (low entropy) but highly redundant: the gzip ratio (compressed size of a sliding text window over its raw size; lower means more repetitive) collapses below τ_{cmp}, forming a distinct low-ratio population despite modest overlap with coherent generations.

2.2 PRM-Guided Stepwise Best-of-N

Best-of-N (Cobbe et al., 2021) samples N candidates and selects the highest-scoring one. Stepwise best-of-N (Snell et al., 2024) repeats this selection at each reasoning step. Given the committed prefix $c_t = (q, s_{<t})$, a policy π samples $\mathcal{S}_t = \{s_t^{(j)} \sim \pi(\cdot \mid c_t)\}_{j=1}^{N}$. A process reward model (PRM) (Uesato et al., 2022; Lightman et al., 2024; Wang et al., 2024a; Yang et al., 2024) then commits $s_t = \arg\max_{s \in \mathcal{S}_t} R(c_t, s)$. Because all candidates share the same prefix, they can be scored in a single batched forward pass.

2.3 Reasoning Model Degeneration

Autoregressive generation can degenerate into off-task or repetitive text instead of completing the response (Xu et al., 2022). Recent work reports that aggressive KV eviction can make such failures more frequent in reasoning models (Cai et al., 2025). We observe two recurring modes in our setting (Figure 2). **Incoherent generation** produces high-entropy sequences of unrelated punctuation, symbols, and mixed-script tokens. **Repetition loops** repeatedly emit the same token or phrase until reaching the length limit. Both modes reduce answer accuracy and waste decoding computation. We target these two discrete failure modes; more gradual failures, such as unnecessarily prolonged but coherent reasoning (Chen et al., 2024; Lotfi et al., 2026), are outside the scope of this work.

3 Method

3.1 Stepwise Interleaved Reasoning

KV-Rescue extends the answer-level complementarity in Figure 1 to stepwise reasoning. At each step, it selects between candidates from a KV-evicted base model and a lightweight full-context helper using a process reward model (PRM).

Setup. Given a question q, a solution is a sequence of steps $\mathbf{s} = (s_1, \ldots, s_T)$ delimited by "$\backslash$n$\backslash$n". We write $c_t = (q, s_{<t})$ for the committed context and $R(c_t, s) \in [0, 1]$ for the PRM score of a

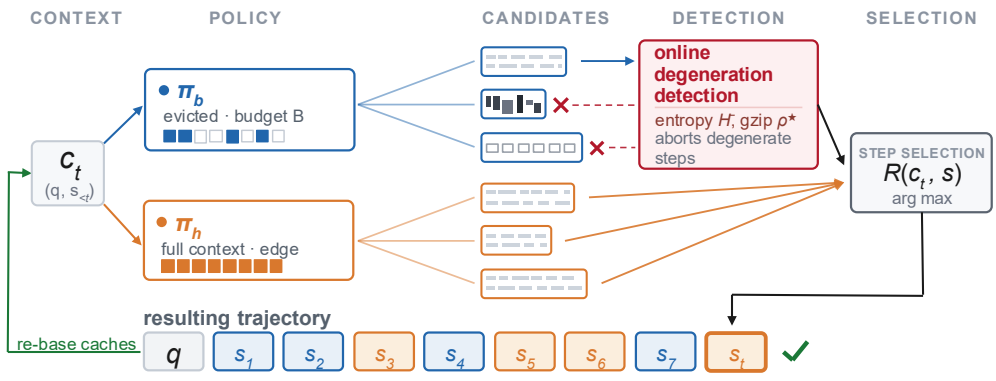


Figure 4: **KV-Rescue.** At each step, candidates are sampled from an evicted base π_b and a full-context helper π_h; an online degeneration detector prunes degenerate base steps, the PRM commits the best step s_t, and base and helper steps interleave. The best-of-N baseline for comparison is in Appendix B.

candidate s. The base model π_b conditions on an evicted cache state $\mathcal{E}_B(c_t)$ with budget B, while the helper π_h conditions on the full context c_t. The models thus share the same trajectory but access different views of its history.

Stepwise interleaving. At step t, KV-Rescue samples base and helper candidate sets and commits the candidate with the highest PRM score:

$$\begin{aligned} \mathcal{C}_t^b &= \left\{ s_t^{b,i} \sim \pi_b(\cdot \mid \mathcal{E}_B(c_t)) \right\}_{i=1}^{N_b}, \\ \mathcal{C}_t^h &= \left\{ s_t^{h,j} \sim \pi_h(\cdot \mid c_t) \right\}_{j=1}^{N_h}, \\ s_t &= \arg\max_{s \in \mathcal{C}_t^b \cup \mathcal{C}_t^h} R(c_t, s). \end{aligned} \tag{2}$$

After committing s_t, the selected fork is retained and the other model processes s_t, so both models share the updated prefix $c_{t+1} = (c_t, s_t)$. The base maintains the evicted cache $\mathcal{E}_B(c_{t+1})$, whereas the helper retains the full context c_{t+1}.

Best-of-$(N_b{+}N_h)$. The one-candidate case uses $N_b{=}N_h{=}1$. Our experiments use $N_b{=}N_h{=}N$, denoted best-of-$N{+}N$, and compare against eviction-only best-of-N with $N_b{=}N$ and $N_h{=}0$. This matches the base-model candidate count and eviction budget; the helper cost is analyzed in the efficiency analysis below.

3.2 Online Degeneration Detection

Under aggressive eviction, base candidates exhibit two recurring failure modes: incoherent generation and repetition loops. Figure 3 shows that they have distinct online signals, which we use to terminate degenerate base candidates before PRM scoring.

Incoherent generation. Incoherent candidates exhibit sustained high predictive entropy. At generated position e, we compute

$$H_e = -\sum_v p_e(v) \log p_e(v). \tag{3}$$

After a guard of G generated tokens, a candidate is terminated when its entropy exceeds τ_{ent} for K consecutive positions:

$$\deg_e^{\text{ent}}(s) = \mathbb{1}\left[e \geq G + K - 1 \ \wedge \ \min_{e-K<r\leq e} H_r > \tau_{\text{ent}} \right]. \tag{4}$$

Repetition loops. Loops are typically low-entropy but highly compressible. Let z_e denote a fixed-width window over the most recent candidate text. Once the window is available, we compute

$$\rho_e = \frac{|\mathrm{gzip}(z_e)|}{|z_e|} \tag{5}$$

and terminate the candidate when $\rho_e < \tau_{\mathrm{cmp}}$. Figure 3(d) reports the per-candidate minimum $\rho^\star(s) = \min_e \rho_e$ for analysis, while online detection uses the current ρ_e.

The detector fires at the first eligible position satisfying either test:

$$\deg_e(s) = \deg_e^{\mathrm{ent}}(s) \vee \mathbb{1}[\rho_e < \tau_{\mathrm{cmp}}]. \tag{6}$$

Only base candidates are screened. With $\deg(s) = \bigvee_e \deg_e(s)$, selection becomes

$$\begin{aligned} \widehat{\mathcal{C}}_t &= \{s \in \mathcal{C}_t^b : \neg\deg(s)\} \cup \mathcal{C}_t^h, \\ s_t &= \arg\max_{s \in \widehat{\mathcal{C}}_t} R(c_t, s). \end{aligned} \tag{7}$$

Helper candidates keep the pool nonempty even if all base candidates are aborted. Both signals are computed online without additional model forward passes.

3.3 KV-Rescue

KV-Rescue combines stepwise interleaving with online degeneration detection (Figure 4), committing the highest-scoring candidate among the surviving base candidates and the full-context helper candidates.

4 Experiments

4.1 Experiment Setup

Models. We use the Qwen2.5-Math family (Yang et al., 2024), which provides reasoners at multiple scales together with a matched PRM. This controlled setup isolates the effect of asymmetric context access without introducing cross-family verifier mismatch. The base is `Qwen2.5-Math-7B-Instruct` or `Qwen2.5-Math-72B-Instruct` (hereafter *7B* and *72B*), the full-context helper is `Qwen2.5-Math-1.5B-Instruct`, and the PRM is `Qwen2.5-Math-PRM-7B` (Zhang et al., 2025). We retain the 7B PRM for the 72B base to test whether a smaller verifier can guide the larger model. We therefore focus the controlled evaluation on mathematical reasoning; Appendix A examines domain coverage through a diagnostic evaluation of available code-domain PRMs. All models run in `bfloat16`, with context capped at 4096 tokens.

Benchmarks and decoding. We report pass@1 on MATH500 Hendrycks et al. (2021), AIME 2024 Zhang & Math-AI (2024), AMC 2023 math ai (2024), GSM8K Cobbe et al. (2021), and OlympiadBench He et al. (2024a), using the official Qwen2.5-Math grader. AIME and AMC are averaged over five seeds, and the remaining benchmarks use one run. We sample at temperature 0.8, top-p 0.95, and top-k 20.

Main results use $N \in \{1, 8\}$. For both methods, N denotes the number of candidates drawn from the base model; KV-Rescue additionally draws N candidates from the 1.5B helper. The comparison matches the base-model candidate budget and eviction budget, rather than the total number of policy candidates. The helper requires approximately 24% and 2.5% of the per-token decode compute of the 7B and 72B bases, respectively.

KV eviction. We use R-KV Cai et al. (2025) with $\lambda = 0.1$ and a protected recent window of 128 tokens. The reported budget $B \in \{64, 128, 256, 512\}$ controls the score-selected entries, giving a physical cache bound of $B' = B + 128$. We also evaluate a no-eviction full-KV reference.

Implementation and hardware. We build on the v1 engine of a modified `vLLM` Kwon et al. (2023). Efficiency and latency are measured on A100 SXM GPUs with 40GB memory. Appendix E provides full implementation details and repeats the 7B evaluation with SnapKV Li et al. (2024), showing that the recovery is not specific to R-KV.

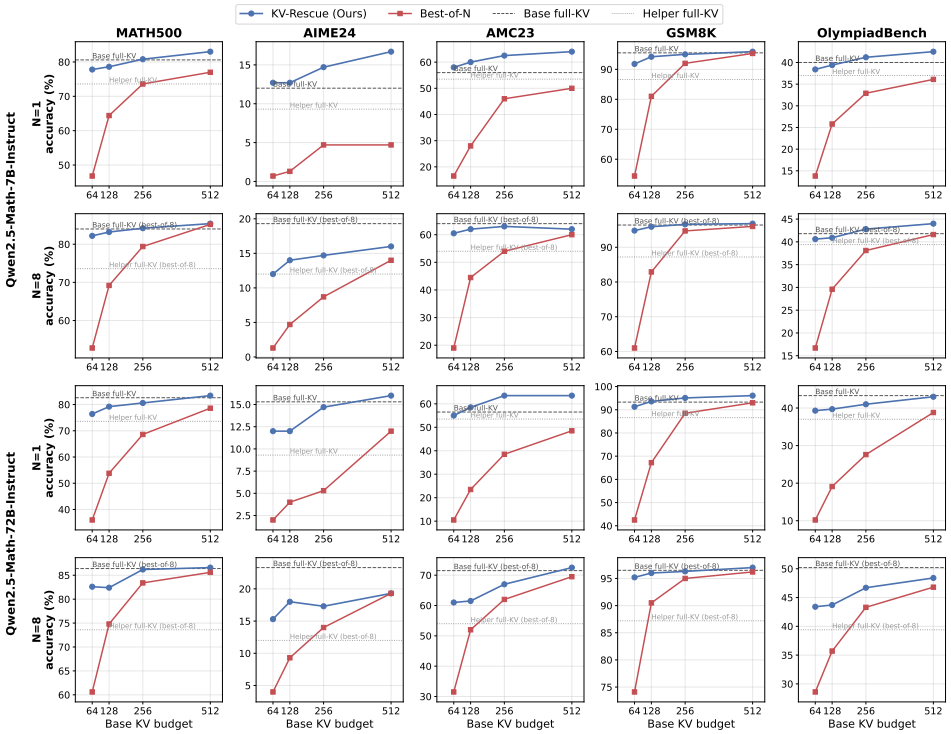


Figure 5: **Main results.** pass@1 vs. base KV budget across the 4×5 grid (Qwen2.5-Math 7B/72B $\times$ N=1, 8; MATH500, AIME24, AMC23, GSM8K, OlympiadBench). Blue is KV-Rescue (ours), red is eviction-only best-of-N; dashed and dotted lines mark the base full-KV and helper full-KV references. Best-of-N falls off sharply as the budget shrinks, while KV-Rescue stays close to base full-KV across budgets.

4.2 Experiment Results

Accuracy versus KV Budget. Figure 5 compares accuracy across the 4×5 grid. Eviction-only best-of-N degrades sharply as B decreases and often approaches or falls below the full-context helper at the smallest budget. KV-Rescue remains substantially more stable and closer to the full-KV reference. At $B = 64$, it recovers an average of 87% of the full-KV accuracy loss across the grid, using $(A_{\text{KVR}} - A_{\text{evict}})/(A_{\text{full}} - A_{\text{evict}})$ for each setting. Increasing N generally improves both methods but does not eliminate the gap under aggressive eviction. The same recovery pattern persists with the 72B base and 7B PRM.

Additional sampling does not compensate for missing context. To test whether the gain comes simply from adding helper candidates, we compare KV-Rescue with $N_b = N_h = 1$ against eviction-only best-of-8. Under aggressive eviction ($B \leq 128$), KV-Rescue outperforms eviction-only best-of-8 for both base sizes on all five benchmarks, despite drawing only one base and one helper candidate per step instead of eight base candidates. Since helper candidates require substantially less computation than base-model candidates, a single low-cost full-context sample is more effective than seven additional evicted-model samples in this regime.

Generation Length. Table 1 compares reasoning-step count, output length, and base-model token generation. As B decreases, eviction-only best-of-N exhibits severe length inflation as degenerate generations approach the length cap. KV-Rescue prevents these runaways and remains closer to the full-KV reference. At $B = 64$ in the 7B, $N = 8$ analysis, it reduces base-model token generation by 43% on average across benchmarks.

Table 1: Decode cost of best-of-N (BoN) vs. KV-Rescue (KVR) on Qwen2.5-Math-7B (N=8): per-problem mean steps, output length, and generated 7B-base / 1.5B-small tokens. Cells are heat-shaded per dataset and metric (green = low, red = high); as the KV budget shrinks the evicted-only BoN runs away (red) while KVR stays controlled (green). *full* = full-KV.

		Steps		Out. len		Base 7B tok		1.5B tok
Dataset	Bud.	BoN	KVR	BoN	KVR	BoN	KVR	KVR
MATH500	*full*	27.0	–	673	–	6,106	–	–
	512	31.8	15.4	670	687	6,189	5,841	6,102
	256	46.7	14.8	701	653	6,704	5,855	5,931
	128	63.8	15.6	754	666	7,607	6,068	6,142
	64	114	21.1	960	671	10,587	6,327	6,560
AIME24	*full*	171	–	1,566	–	16,803	–	–
	512	148	43.8	1,510	1,369	16,457	12,775	13,591
	256	222	41.9	1,663	1,295	18,943	12,842	13,620
	128	275	51.1	1,768	1,362	20,660	13,159	14,307
	64	229	64.9	1,600	1,388	20,573	13,584	15,812
AMC23	*full*	40.4	–	966	–	9,377	–	–
	512	35.2	25.0	959	1,011	9,695	9,574	9,476
	256	73.6	23.4	1,000	988	10,440	9,159	10,350
	128	121	30.7	1,166	1,015	12,706	9,654	10,451
	64	152	32.1	1,295	944	14,947	9,209	9,978
GSM8K	*full*	8.3	–	307	–	2,477	–	–
	512	8.8	6.6	307	303	2,500	2,448	2,317
	256	9.9	6.8	314	306	2,577	2,484	2,380
	128	28.9	7.0	376	307	3,422	2,537	2,446
	64	96.2	8.5	619	309	7,056	2,607	2,590
OlympiadBench	*full*	57.5	–	1,016	–	9,499	–	–
	512	84.9	26.8	1,092	1,049	11,040	9,550	10,189
	256	98.8	29.1	1,113	1,026	11,673	9,458	10,242
	128	139	36.2	1,233	1,053	13,793	9,819	13,132
	64	187	36.4	1,425	1,040	17,327	10,171	10,920

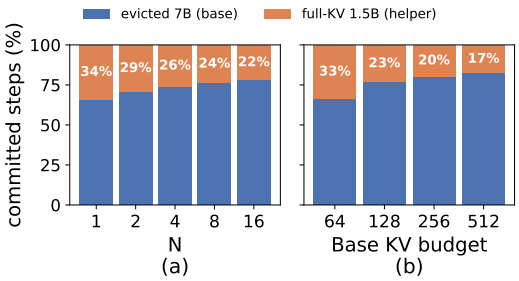


Figure 6: Share of committed steps selected from the evicted 7B base and the full-context 1.5B helper on MATH500 with best-of-N+N decoding: **(a)** as N varies at budget 128 and **(b)** as the KV budget varies at N=8.

5 Discussion

5.1 Recovering KV-Eviction-Induced Information Loss

We examine how the two models contribute to the recovered trajectory. Figure 6 shows that the helper is not merely a rare fallback: it supplies 17–34% of committed steps across the evaluated settings. Its share increases from 17% at budget 512 to 33% at budget 64, consistent with full-context candidates becoming more useful as the base loses more history. The base nevertheless contributes most steps, and the source ratio changes little as N grows, indicating that KV-Rescue combines the two models rather than replacing the base with the helper.

Figure 9 in Appendix D shows a representative case in which KV-Rescue succeeds by interleaving steps from two models that fail independently.

5.2 Degeneration Analysis

Table 2 isolates the contributions of interleaving and early exit. At budget 64, eviction-only best-of-N raises the degeneration rate from 2.4% under full-KV to 13.4%. Sampling alone does not prevent these runaways because every candidate is generated from the same evicted cache.

Table 2: Ablation of interleaving and early exit on MATH500 (N=8, Qwen2.5-Math-7B). Rows are cumulative. *Degen.* is the fraction of incorrect trajectories with at least 256 committed steps, used to identify context-length runaways. *Base tok.* is the mean number of tokens generated by base-model candidates per problem; *Steps* is the mean number of committed steps; and *EE* is the fraction of base candidates terminated by early exit.

Budget	Method	Acc. (%)	Degen. (%)	Base tok.	Steps	EE (%)
full	full-KV	84.0	2.4	6,106	27	–
64	best-of-N	52.8	13.4	10,587	114	–
	+ interleaving	81.8	2.8	7,898	36	–
	+ early exit	82.2	0.2	6,327	21	8.0
128	best-of-N	69.2	6.0	7,607	64	–
	+ interleaving	81.2	1.4	6,673	25	–
	+ early exit	83.2	0.0	6,068	16	6.4

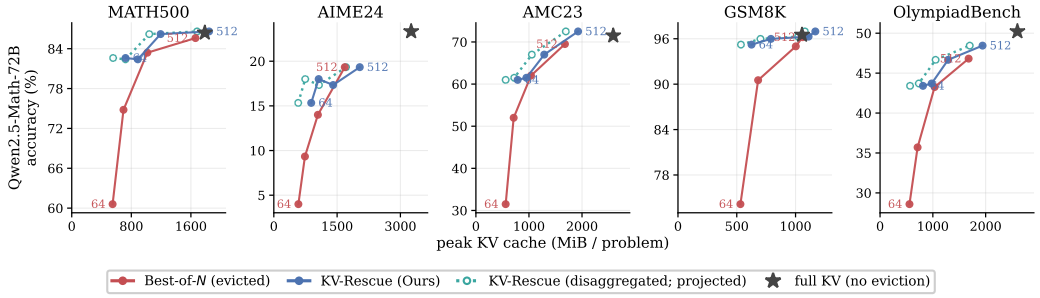


Figure 7: **Accuracy versus peak KV-cache memory.** Results for Qwen2.5-Math-72B with N=8 at equal generation length. Point labels denote the base KV budget; the dotted curve projects a disaggregated deployment with the helper KV on a separate accelerator.

Interleaving substantially reduces trajectory degeneration. Adding full-context helper candidates gives the PRM alternatives that do not share the base model's missing context. At budget 64, interleaving reduces degeneration from 13.4% to 2.8%, lowers the mean trajectory from 114 to 36 steps, and raises accuracy from 52.8% to 81.8%. This accounts for most of the recovery.

Early exit removes candidate-level token waste. Interleaving can avoid committing a degenerate base candidate while still decoding that candidate to the step limit. Early exit terminates such branches during generation. At budget 64, it fires on 8.0% of base candidates and reduces base-model token generation by a further 20% (7.9k→6.3k) without reducing accuracy. It also lowers degeneration from 2.8% to 0.2%, while eliminating it under the table's criterion at budget 128. Appendix F further supports this accuracy-preserving efficiency gain: distribution-based calibration and threshold analysis show that the detector maintains high recall with near-zero false-positive rates.

5.3 Efficiency Analysis

Appendix G analyzes KV memory and per-token decode compute as context grows. At a 4k context, KV-Rescue uses 0.31 GiB of KV memory in the 72B setting, compared with 1.25 GiB for full-KV, while the 1.5B helper adds approximately 2.5% of the 72B base model's per-token compute.

Accuracy–memory trade-off. Figure 7 shows the resulting accuracy–memory trade-off. At matched peak KV-cache memory, KV-Rescue outperforms eviction-only best-of-N across all five benchmarks even when the helper is co-located. The dotted curve additionally projects the base-device memory footprint under a disaggregated deployment, assuming that the independent 1.5B helper runs on a separate accelerator.

Table 3: Wall clock efficiency (Qwen2.5-Math-72B, N=8, KV budget 64; optimized engine with batched eviction and dynamic KV reserve). *prob/min*: steady-state throughput (excludes one-time load); t_{gen}: per-problem generation time (base π_b and full-context helper π_h run concurrently); t_{score}: PRM time; all amortized over batch concurrency. Max length 4096; full-KV OOMs at batch 64.

Method	Batch	prob/min	t_{gen} (s)	t_{score} (s)	72B tok
full-KV	8	6.7	8.41	0.375	5,392
	64		*OOM*		
BoN	8	1.2	44.45	1.619	18,144
	64	3.2	12.84	0.193	18,371
KV-Rescue	8	4.2	12.67	0.410	5,951
	64	10.6	4.58	0.034	5,460

End-to-end efficiency. Table 1 shows that KV-Rescue reduces base-model token generation by preventing runaway degeneration. Table 3 reports end-to-end performance on the same eight A100 40GB GPUs, including helper generation, PRM scoring, and system overhead. At batch 64, KV-Rescue achieves 10.6 problems/min versus 3.2 for eviction-only best-of-N, while reducing base generation by 70% (18.4k$\rightarrow$$5.5$k tokens); full-KV cannot fit at this batch size.

6 Related Work

KV cache compression. A large body of work reduces KV-cache memory through quantization Hooper et al. (2024); Liu et al. (2024); Son et al. (2025), merging Wang et al. (2024b), and eviction Zhang et al. (2023); Li et al. (2024); recent work further adapts eviction to reasoning models Cai et al. (2025). These methods operate within a single model, reducing the precision, redundancy, or number of entries in its cache. Closest to our setting is SmallKV Zhao et al. (2026), which maintains a full-cache small model and uses its attention to guide large-model eviction and approximate the contribution of marginal tokens. The large model nevertheless remains the sole generator. KV-Rescue instead treats the small full-context model as an independent reasoning policy and interleaves its steps with those of the evicted base. It operates above the eviction mechanism and can be orthogonally combined with different eviction policies.

Test-time and multi-model inference. Test-time scaling improves reasoning by sampling and selecting candidates through best-of-N Cobbe et al. (2021) and PRM-guided step or beam search Snell et al. (2024); Lightman et al. (2024). These methods typically sample from a single policy conditioned on the same context state. KV-Rescue instead draws step candidates from two policies with asymmetric context access, using selection to recover information unavailable to the evicted base. Speculative decoding similarly combines small and large models, but uses the small model to draft tokens while preserving the large model's output distribution Leviathan et al. (2023); Chen et al. (2023). In KV-Rescue, either model may contribute a committed reasoning step, and the goal is to recover reasoning quality under KV eviction rather than accelerate an unchanged target policy.

7 Conclusion

KV eviction bounds long-reasoning memory but may discard information needed later. We showed that much of this loss is an information rather than a capability gap: an evicted large model and a lightweight full-context model make complementary errors. KV-Rescue exploits this complementarity without training by interleaving their reasoning steps and terminating degenerate base candidates online. Across five math benchmarks with Qwen2.5-Math 7B and 72B, it recovers 87% of the accuracy lost at budget 64 and reduces base-model generation by 43% in our decode-cost analysis. At batch 64, it achieves $3.3\times$ the throughput of eviction-only best-of-N, while full-KV runs out of memory. These results show that complementary context views can preserve reasoning under tight KV-memory constraints.

AI USE STATEMENT

In this work, we used generative AI tools for method implementation and assistance with experimental setup and execution. We have not used generative AI tools for any other tasks requiring disclosure, and the remaining required-disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for software debugging and for language editing and polishing of the manuscript. We have reviewed all AI-assisted work. LLM-generated code was reviewed, verified, and tested for correctness by two authors, and all reported results were checked against the corresponding experimental outputs. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We document the components required to reproduce our results throughout the paper and appendix. Sections 3.1 and 3.2 formally specify stepwise interleaving and online degeneration detection, and Algorithm 1 provides the complete decoding procedure. Section 4.1 describes the experimental protocol, while Appendix E details the models, decoding parameters, eviction configuration, detector thresholds, inference engine, and hardware setup. Appendix F reports detector calibration and false-positive analysis. Appendix G provides the assumptions and derivations used in the memory and compute analysis. All evaluated models and datasets are publicly available and identified in the paper. Upon acceptance, we will publicly release our implementation, experiment configurations, and evaluation scripts needed to reproduce the reported results.

REFERENCES

Zefan Cai, Wen Xiao, Hanshi Sun, Cheng Luo, Yikai Zhang, Ke Wan, Yucheng Li, Yeyang Zhou, Li-Wen Chang, Jiuxiang Gu, Zhe Dong, Anima Anandkumar, Abedelkadir Asi, and Junjie Hu. R-kv: Redundancy-aware kv cache compression for training-free reasoning models acceleration. *arXiv preprint arXiv:2505.24133*, 2025.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do not think that much for 2+3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, 2024a.

Jujie He, Tianwen Wei, Rui Yan, Jiacai Liu, Chaojie Wang, Yimeng Gan, Shiwen Tu, Chris Yuhao Liu, Liang Zeng, Xiaokun Wang, Boyang Wang, Yongcong Li, Fuxiang Zhang, Jiacheng Xu, Bo An, Yang Liu, and Yahui Zhou. Skywork-o1 open series, November 2024b. URL `https://doi.org/10.5281/zenodo.16998085`.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *arXiv preprint arXiv:2401.18079*, 2024.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph 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*, pp. 611–626, 2023.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

Qingyao Li, Xinyi Dai, Xiangyang Li, Weinan Zhang, Yasheng Wang, Ruiming Tang, and Yong Yu. CodePRM: Execution feedback-enhanced process reward model for code generation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 8169–8182, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.428. URL https://aclanthology.org/2025.findings-acl.428/.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*, 2024.

Sanae Lotfi, Polina Kirichenko, Steven Li, and Zechun Liu. Quantized reasoning models think they need to think longer, but they do not. *arXiv preprint arXiv:2606.00206*, 2026.

math ai. amc23: A benchmark dataset for american mathematics competitions 2023. https://huggingface.co/datasets/math-ai/amc23, 2024.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Donghyun Son, Euntae Choi, and Sungjoo Yoo. Nsnquant: A double normalization approach for calibration-free low-bit vector quantization of kv cache. *Advances in Neural Information Processing Systems*, 38:43124–43159, 2025.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024a.

Zheng Wang, Boxiao Jin, Zhongzhi Yu, and Minjia Zhang. Model tells you where to merge: Adaptive kv cache merging for llms on long-context tasks. *arXiv preprint arXiv:2407.08454*, 2024b.

Jin Xu, Xiaojiang Liu, Jianhao Yan, Deng Cai, Huayang Li, and Jian Li. Learning to break the loop: Analyzing and mitigating repetitions for neural text generation. *Advances in Neural Information Processing Systems*, 35:3082–3095, 2022.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2024, 2024. URL https://huggingface.co/datasets/math-ai/aime24.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10495–10516, 2025.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2306.14048.

Yi Zhao, Yajuan Peng, Nguyen Cam-Tu, Zuchao Li, Xiaoliang Wang, Hai Zhao, and Xiaoming Fu. Smallkv: Small model assisted compensation of kv cache compression for efficient llm inference. *Advances in Neural Information Processing Systems*, 38:48637–48665, 2026.

A Domain Coverage and PRM Availability

KV-Rescue requires a process reward model (PRM) that can reliably rank candidate steps. To our knowledge, Qwen2.5-Math was the only open model family available at the time of our experiments with an official, in-domain PRM matched to the policy family and its step format. This allows us to evaluate KV-Rescue without training an additional reward model.

Code-domain attempt. To determine whether a controlled code-domain evaluation is feasible with currently open models, we configure KV-Rescue with `Qwen2.5-Coder-7B-Instruct` as the evicted base and `Qwen2.5-Coder-1.5B-Instruct` as the helper. Because Qwen2.5-Coder has no official family-matched PRM, we consider two code-capable PRMs released post hoc by third parties: CodePRM (Li et al., 2025) and `Skywork-o1-Open-PRM-Qwen-2.5-7B` (He et al., 2024b). Before running a full KV-budget sweep, we test whether these models provide a sufficiently reliable selection signal by measuring pass/fail AUROC on full-KV HumanEval trajectories generated by `Qwen2.5-Coder-7B-Instruct`. We apply the same diagnostic to `Qwen2.5-Math-PRM-7B` on MATH500 trajectories generated by `Qwen2.5-Math-7B-Instruct`. CodePRM and the math PRM are evaluated with their corresponding policy families; Skywork uses a Qwen2.5-Math-7B backbone but scores the same code trajectories as CodePRM.

Table 4: **Outcome discrimination of the PRM scores.** AUROC for separating passing from failing full-KV trajectories using the last, minimum, or mean step score. "Policy" denotes the model that generated the evaluated trajectories. MATH500 contains 500 trajectories (420 pass); HumanEval contains 492 trajectories over three seeds (425 pass). Skywork uses a Qwen2.5-Math-7B backbone.

PRM	Policy (set)	Last	Min	Mean
CodePRM	Qwen2.5-Coder-7B (HumanEval)	0.643	0.561	0.613
Skywork-o1-PRM-7B	Qwen2.5-Coder-7B (HumanEval)	0.686	0.712	0.730
Qwen2.5-Math-PRM-7B	Qwen2.5-Math-7B (MATH500)	**0.848**	**0.911**	**0.910**

Although outcome-level AUROC does not directly measure within-step candidate ranking, the code PRMs show substantially weaker full-trajectory discrimination: their mean-score AUROCs are 0.613 and 0.730, compared with 0.910 for the math PRM. We therefore did not regard the available code PRMs as sufficiently reliable selectors for a controlled evaluation of the KV-Rescue mechanism.

Accordingly, our restriction to mathematics reflects the availability of a sufficiently accurate, family-matched PRM rather than a math-specific assumption in KV-Rescue. Its core mechanism applies unchanged across domains, but validating it requires a reliable PRM for the target policy and domain. We therfore restrict our quantitative evaluation to the controlled setting with a reliable, family-matched PRM, and defer broader evaluation until comparably capable PRMs become available.

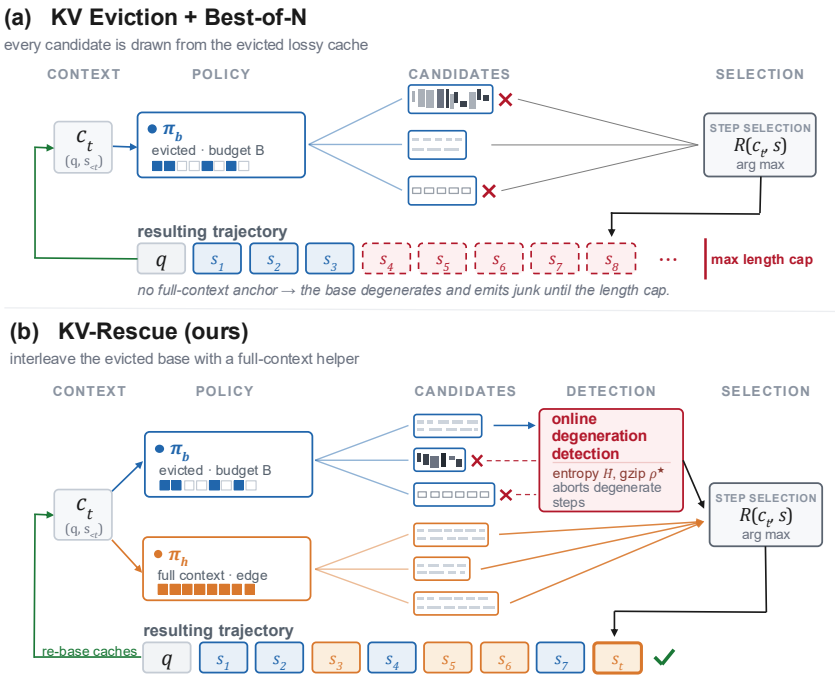


Figure 8: **KV-Rescue. (a)** Best-of-N under KV eviction draws every candidate from the same lossy cache, so the information lost to eviction leads the base to produce wrong or degenerate steps. **(b)** KV-Rescue samples steps from an evicted base π_b and a full-context helper π_h; an online degeneration detector prunes degenerate base steps, the PRM commits the best step s_t, and base and helper steps interleave.

B KV-Rescue vs KV Evicted Best-of-N

Figure 8 contrasts the two settings. Under KV eviction, best-of-N draws every candidate step from the same lossy cache, so all candidates share the base model's missing context: the PRM can only select among them, and sampling cannot recover the evicted information. As a result the base answers incorrectly or degenerates, emitting incoherent or repetitive steps until the length cap (Fig. 8a).

KV-Rescue instead samples candidate steps from two policies at each step, the evicted base π_b and a full-context helper π_h. An online degeneration detector prunes degenerate base candidates before selection, the PRM commits the best surviving step s_t, and both caches re-base onto c_{t+1}. Committed base and helper steps interleave over the solution, each covering the other's failure and recovering the correct trajectory (Fig. 8b).

C KV-Rescue Pseudocode

Algorithm 1 summarizes one decoding pass. Each step samples N_b candidates from the evicted base π_b (on the compressed cache $\mathcal{E}_B(c)$) and N_h from the full-context helper π_h, aborting any that fire the online detector $\deg(\cdot)$ mid-generation. The PRM commits the best non-degenerate step s_t, which extends the context c; both caches then re-base onto the shared prefix. The loop repeats until an `<eos>` step or the length cap L.

D Qualitative Example of Stepwise Interleaving

Figure 9 illustrates how the two models provide complementary information. The evicted base loses track of the requested quantity, while the full-context helper makes a derivation error; interleaving their steps allows KV-Rescue to reach the correct answer.

Algorithm 1 KV-Rescue decoding for one question

Require: Question q; evicted base policy π_b with budget B
Require: Full-context helper policy π_h; PRM R
Require: Candidate counts N_b, N_h, where $N_h \geq 1$
Require: Thresholds $\tau_{\text{ent}}, \tau_{\text{cmp}}$; length cap L
Ensure: Solution $\mathbf{s}$

$c \leftarrow q$; $\mathbf{s} \leftarrow ()$; $t \leftarrow 1$
$K_b \leftarrow \mathcal{E}_B(q)$; $K_h \leftarrow q$
while $\text{len}_{\text{tok}}(\mathbf{s}) < L$ **do**
  Sample $s_t^{b,i} \sim \pi_b(\cdot \mid K_b)$, $i = 1, \ldots, N_b$
  Collect the candidates in $\mathcal{C}_t^b$; abort when $\deg(s_t^{b,i}) = 1$
  Sample $s_t^{h,j} \sim \pi_h(\cdot \mid K_h)$, $j = 1, \ldots, N_h$
  Collect the candidates in $\mathcal{C}_t^h$
  $\widehat{\mathcal{C}}_t^b \leftarrow \{s \in \mathcal{C}_t^b : \neg\deg(s)\}$
  $\mathcal{C}_t \leftarrow \widehat{\mathcal{C}}_t^b \cup \mathcal{C}_t^h$
  $s_t \leftarrow \arg\max_{s \in \mathcal{C}_t} R(c, s)$
  $\mathbf{s} \leftarrow \mathbf{s} \| s_t$; $c \leftarrow (c, s_t)$
  if $s_t \in \widehat{\mathcal{C}}_t^b$ **then**
    Retain the selected base fork as K_b
    Append s_t to K_h
  else
    Retain the selected helper fork as K_h
    Append s_t to K_b and apply $\mathcal{E}_B$
  end if
  if s_t ends with <eos> **then**
    break
  end if
  $t \leftarrow t + 1$
end while
return $\mathbf{s}$

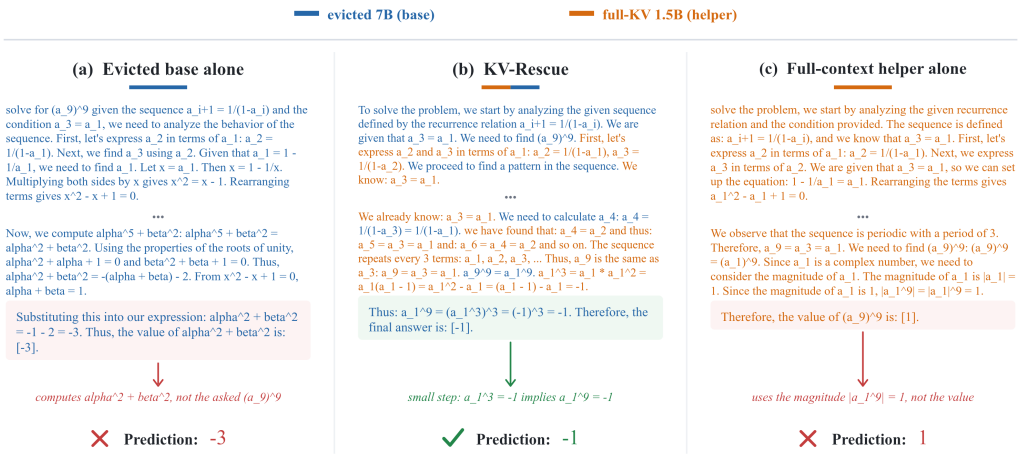


Figure 9: **A representative example solved by KV-Rescue but not by either model independently. (a)** Evicted base, **(b)** KV-Rescue, and **(c)** full-context helper. Blue and orange denote committed base and helper steps, respectively.

E Implementation Details

Models. We instantiate KV-Rescue on the Qwen2.5-Math family (Yang et al., 2024). The *base* reasoner is `Qwen2.5-Math-7B-Instruct` (and `Qwen2.5-Math-72B-Instruct` for the

scaling study); the full-KV *helper* is `Qwen2.5-Math-1.5B-Instruct`. Step selection uses the process reward model `Qwen2.5-Math-PRM-7B` (Zhang et al., 2025), a token-classification reward model: we form prompt $\oplus \backslash n\backslash n \oplus s_1 \oplus \cdots \oplus s_k \oplus \backslash n\backslash n$ and take the softmax probability of the positive label at the final token as the step score $R(\cdot)$ (the step separator is $\backslash n\backslash n$, matching the PRM's training-time separator). All models run in `bfloat16`.

KV eviction. Eviction is applied to the base model only, via R-KV (Cai et al., 2025), which every step compresses the base's KV to a fixed budget B by scoring cached tokens with a mix of query-attention importance and key redundancy (cosine similarity), keeping the top-B plus a small recent window. We use budgets $B \in \{64, 128, 256, 512\}$, a headroom buffer w=128 (so the physical KV is bounded by B'=B+w), and mix weight λ=0.1. The helper and PRM keep their full KV cache.

Best-of-$(N$+$N)$ trajectory. We implement KV-Rescue on vLLM v1 (Kwon et al., 2023) using a persistent sequence that stores the accepted trajectory's evicted base KV. At each step, the engine (i) forks N base candidates by copying the live post-eviction KV blocks, (ii) generates N full-context helper candidates in parallel, and (iii) scores all $2N$ candidates with the PRM and commits the highest-scoring one. If a base candidate is selected, its fork becomes the new persistent state. If a helper candidate is selected, the base encodes the selected step in a single forward pass and then applies eviction. The base state is therefore updated incrementally, without backtracking or re-prefilling the accepted history. We evaluate $N \in \{1, 8\}$.

For throughput, the engine uses event-driven continuous batching across problems. Base candidates from multiple problems share a running set and are advanced token by token by vLLM. Each problem proceeds to PRM scoring as soon as its candidates reach the step delimiter and is replaced independently when completed, avoiding synchronization on slower problems. Helper generation and PRM scoring run asynchronously on background threads and overlap with base decoding.

Online degeneration detection. Each base candidate is screened during generation. The entropy rule terminates a candidate when its predictive entropy exceeds τ_{ent}=7.0 for K=12 consecutive tokens after a guard of G=24 tokens. The compression rule terminates repetitive generation when the gzip ratio of a sliding text window of width w_{cmp}=800 falls below τ_{cmp}=0.18. Both signals are computed online without additional model forward passes.

Decoding and evaluation. Following the Qwen2.5-Math non-thinking regime we sample at temperature 0.8, top-p 0.95, top-k 20, with maximum context 4096, up to 1024 reasoning steps, and at most 512 tokens per step. We evaluate pass@1 accuracy on MATH500, AIME24, AMC23, GSM8K, and OlympiadBench, extracting the final `\boxed` answer with the official Qwen2.5-Math grader; AIME24 and AMC23 are averaged over five seeds and the rest use a single seed.

Serving. The base engine, PRM, and helper run as separate vLLM instances; the PRM and helper are queried over asynchronous HTTP, allowing their execution to overlap with base decoding. For the 7B base we co-locate all three on each GPU and shard the problem set across GPUs with work-stealing; for the 72B base we shard the base model with tensor parallelism (TP=8) and co-locate the PRM (TP=4) and helper alongside it.

F Detector Calibration and False Positives

Scope. The detector is not intended to identify semantic errors such as goal drift or an incorrect but fluent derivation. Such candidates cannot be reliably identified from inexpensive token-level statistics and may still recover in later steps, making early termination potentially harmful. Semantic quality is instead handled by the PRM after candidate steps are formed. The detector targets incoherent generation and repetition, which exhibit clear online signals and otherwise tend to continue toward the generation limit.

Calibration. The thresholds are selected once from the distributions in Figure 3, rather than by maximizing downstream accuracy. For entropy, we analyze 11,960 evicted-base candidates from MATH500 runs with $B \in \{64, 128\}$ and N=8. The normal and incoherent modes are nearly disjoint: only 3.0% of eligible candidates lie between the low-entropy mode in $[0, 1)$ nats and the

incoherent mode in $[11, 12)$ nats, while useful candidates remain below 3.44 nats. We therefore set τ_{ent}=7.0 inside this gap. For compression, we analyze 1,332 `Qwen2.5-Math-7B` generations, including 97 capped repetition loops. The distributions overlap modestly because correct derivations may reuse algebraic structure, but loops still form a distinct low-ratio population: at w_{cmp}=800, their median compression ratio is 0.020, compared with a fifth percentile of 0.347 for correct generations. We use the larger window to capture longer-period repetitions and set τ_{cmp}=0.18 conservatively below the correct population. The same $(\tau_{\text{ent}}, K, G, \tau_{\text{cmp}}, w_{\text{cmp}}) = (7.0, 12, 24, 0.18, 800)$ is used across all reported settings.

False positives. Table 5 reports offline detector replay at the deployed thresholds. The entropy false-positive count remains zero throughout $\tau_{\text{ent}} \in [4, 11.5]$ and across 45 combinations of G, K, and τ_{ent}, for which recall stays between 85.0% and 86.1%. Together with the distributional separation in Figure 3, these results support a conservative operating point for both rules.

Table 5: **Detector validation at the deployed thresholds.** For entropy, the false-positive rate counts detector-triggered candidates that would otherwise be selected by the PRM and lead to a correct final trajectory; for compression, it counts eligible correct full-KV trajectories that trigger the rule. Recall is measured on length-capped candidates and capped repetition loops, respectively.

Detector	FP rate	Recall
Entropy	0.00%	85.0%
Compression	0.12%	87.6%

The end-to-end ablation in Table 2 is consistent with this conservative behavior. At B=64, early exit reduces base-model generation by 20% and lowers the runaway rate from 2.8% to 0.2% without reducing accuracy; at B=128, it eliminates runaways under the reported criterion.

G Overhead Derivation

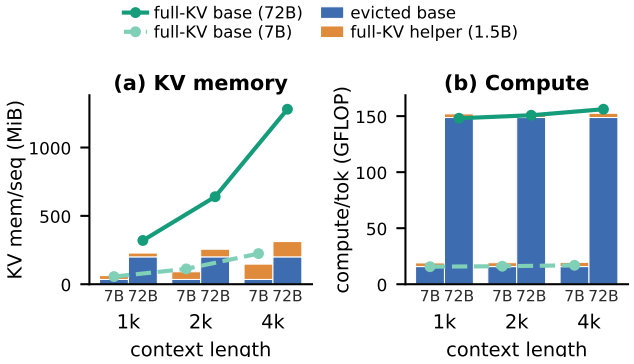


Figure 10: **Per-sequence KV memory (a) and per-token decode compute (b) of KV-Rescue vs. context length**, for a 7B and a 72B base (bf16, R-KV budget 512, B'=640). Bars are KV-Rescue: the *evicted base* (blue, bounded by the budget) plus the *full-KV helper* 1.5B (orange). Lines are the *full-KV base* baseline.

We derive the per-sequence KV memory and per-token decode compute shown in Figure 10, and the context length at which evicted decode becomes cheaper than full-KV, and evaluate them with the Qwen2.5-Math constants.

G.1 Notation

For a decoder-only transformer let L be the number of layers, d the model dimension, h_{kv} the number of key/value heads, d_h the head dimension, P the parameter count, and b the bytes per

element (b=2 for `bfloat16`). Let ℓ be the current context length (tokens), B the R-KV budget, w the eviction buffer, and $B'=B+w$ the resulting physical KV bound. Table 6 lists the values we use.

Table 6: Model constants used in the derivation (from each model's `config.json`).

model	L	d	h_{kv}	d_h	d_{ffn}	P
base 7B	28	3584	4	128	18944	7.62×10^9
base 72B	80	8192	8	128	29568	72.7×10^9
helper 1.5B	28	1536	2	128	8960	1.54×10^9

G.2 KV MEMORY

Each cached token stores a key and a value vector of size d_h in every KV head of every layer, so the per-token, per-sequence KV footprint is

$$m_{\text{kv}} \;=\; 2\,L\,h_{\text{kv}}\,d_h\,b, \tag{8}$$

the leading 2 counting keys and values. Substituting Table 6 gives $m_{\text{kv}}^{\text{7B}} = 56\,\text{KiB}$, $m_{\text{kv}}^{\text{72B}} = 320\,\text{KiB}$, and $m_{\text{kv}}^{\text{1.5B}} = 28\,\text{KiB}$ per token.

Full-KV memory grows linearly in context, whereas eviction bounds it at the budget:

$$M_{\text{full}}(\ell) = m_{\text{kv}}\,\ell, \qquad M_{\text{evict}} = m_{\text{kv}}\,B' \quad (\text{constant in } \ell). \tag{9}$$

The eviction saving on the base model is therefore *model-size independent*,

$$\frac{M_{\text{full}}(\ell)}{M_{\text{evict}}} = \frac{\ell}{B'}, \tag{10}$$

e.g. ℓ/B'=8192/640 $\approx$ 12.8× at ℓ=8k, B=512. For the full method a problem generates N candidate sequences; the base is evicted and the helper keeps full KV, so

$$\begin{aligned} M_{\text{KVR}} &= N\big(m_{\text{kv}}^{\text{base}}B' + m_{\text{kv}}^{\text{help}}\,\ell\big), \\ M_{\text{full-BoN}} &= N\,m_{\text{kv}}^{\text{base}}\,\ell. \end{aligned} \tag{11}$$

The helper term $m_{\text{kv}}^{\text{help}}\ell$ does not depend on the base model, so it dominates M_{KVR} when the base is small and is negligible when the base is large. At ℓ=4k, B=512, N=8, the ratio $M_{\text{full-BoN}}/M_{\text{KVR}}$ is $1.75/1.15 \approx 1.5\times$ for the 7B base but $10.0/2.44 \approx 4.1\times$ for the 72B base.

G.3 DECODE COMPUTE

For one decoded token the forward cost splits into a context-independent dense part (all linear projections and the FFN, $\approx 2P$ FLOP by the standard multiply–add count) and an attention part that is linear in the number of cached tokens attended to:

$$C(\ell) = \underbrace{2P}_{C_{\text{dense}}} \;+\; \underbrace{4\,L\,d\,\ell}_{C_{\text{attn}}(\ell)}. \tag{12}$$

The attention term is the two matmuls per layer, $QK^\top$ and the value aggregation; each runs over the ℓ cached positions across the full query width d, costing $d\ell$ multiply–adds per token, so at two FLOP per multiply–add the two matmuls give $4d\ell$ FLOP, summed over L layers. (Under grouped-query attention the KV *memory* of Eq. 8 scales with the h_{kv} cached KV heads, whereas attention *compute* scales with the full query width d.) With Table 6, C_{dense} is 15.2 and 145.4 GFLOP/token for 7B and 72B, while $C_{\text{attn}}(\ell) = 0.401\text{M} \cdot \ell$ and $2.62\text{M} \cdot \ell$ FLOP respectively; at ℓ=4k attention is only 11% (7B) / 7% (72B) of the forward, so bounded-KV attention saves little *compute* at short context.

Eviction is not free, however. When R-KV fires (every step once $\ell > B'$), it makes, per layer, one attention-score pass plus a key-similarity, pooling and top-k selection over the B' physical tokens; to first order this is $\alpha\,C_{\text{attn}}(B')$ with $\alpha \in [1.5, 3]$, so the evicted per-token cost

$$C_{\text{evict}} = C_{\text{dense}} + (1{+}\alpha)\,C_{\text{attn}}(B') = 2P + (1{+}\alpha)\,4LdB' \tag{13}$$

depends only on the bounded B' and is thus flat in ℓ. Evicted decode is cheaper per token once the full-context attention outgrows this bounded cost, $C_{\text{attn}}(\ell) \geq (1+\alpha)\, C_{\text{attn}}(B')$. As C_{attn} is linear in ℓ, this happens beyond a crossover that is independent of model size,

$$\ell^\star = (1+\alpha)\, B'. \tag{14}$$

We measure this crossover at $\ell^\star \approx 2500$; for B'=640 that corresponds to $\alpha \approx 2.9$, near the top of the $[1.5, 3]$ range and consistent with R-KV doing more than a single attention pass (it also computes the key similarity, pooling and top-k). Below $\ell^\star$ eviction is slightly slower, as the scoring outweighs the attention it saves; above it, the theoretical projection predicts a widening gap, reaching about 2% at 4k and 6% at 12k. The N helper candidates add, per token, a fraction $\rho = C^{\text{help}}(\ell)/C^{\text{base}}_{\text{evict}}$ of a base candidate,

$$\rho = \frac{2P^{\text{help}} + 4L^{\text{help}} d^{\text{help}} \ell}{2P^{\text{base}} + (1+\alpha)\, 4L^{\text{base}} d^{\text{base}} B'}; \tag{15}$$

the helper is a fixed 1.5B model, so this fraction falls as the base grows, from about 24% for the 7B base to 2.5% for the 72B base at ℓ=4k.

These are the quantities in Figure 10: eviction caps the base KV cache (an ℓ/B' saving, Eq. 10) and its attention cost, at the price of a per-token overhead that is only repaid past $\ell^\star \approx 2500$ (Eq. 14), plus the small helper cost ρ (Eq. 15) of roughly 24% for the 7B base and 2.5% for the 72B base.

H Experiment Results

H.1 KV-Rescue with SnapKV

KV-Rescue is agnostic to the underlying evictor: it treats the base as a black-box policy conditioned on whatever cache the evictor keeps. To verify this, Figure 11 repeats the 7B study with R-KV replaced by SnapKV (Li et al., 2024), and the same pattern holds, showing the recovery does not depend on the specific eviction method.

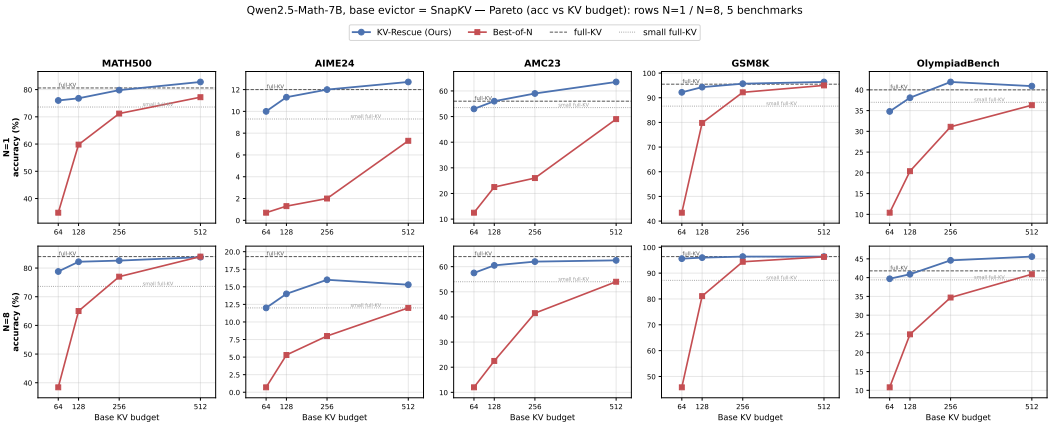


Figure 11: **KV-Rescue with SnapKV eviction (Qwen2.5-Math-7B).** pass@1 vs. base KV budget over the five benchmarks (MATH500, AIME24, AMC23, GSM8K, OlympiadBench), for N=1 (top) and N=8 (bottom). Blue is KV-Rescue (ours), red is eviction-only best-of-N; dashed and dotted lines mark the base full-KV and helper (small) full-KV references. Same setup as main results with R-KV replaced by SnapKV.