SPARSE ATTENTION WITH ENTROPY-TRIGGERED KV REPAIR

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse attention reduces the per-step decoding cost of language models trained with full attention. However, it can lengthen generations and lower accuracy, offsetting the per-step savings with additional tokens. We find that sparse-attention errors are most detrimental at steps where the full-attention next-token entropy is high: applying sparse attention only at these steps degrades accuracy far more than applying it at an equal number of random steps. We further find that at these high-entropy steps, most of the degradation stems from errors accumulated into the key-value (KV) cache during earlier sparse decoding, so even attending to the entire cache at such steps results in performance degradation. Motivated by these findings, we propose HERA (**H**igh-**E**ntropy-triggered KV **R**ep**A**ir), a method which uses sparse next-token entropy to trigger a KV repair — a full-attention pass that recomputes and replaces the cached KV states to yield the dense next-token distribution for sampling. We evaluate HERA on three reasoning models across four benchmarks. On MATH-500 with Qwen3-8B, HERA improves accuracy by 18.8 percentage points over the sparse-decoding baseline (i.e., Quest) with 25.7% fewer tokens, and by 5.6 points over a KV repair at fixed intervals baseline (i.e., ReSA) with 7.4% fewer tokens and 13.7% fewer repair calls.

1 INTRODUCTION

Large Language Models solve hard problems by generating long chains of thought (Wei et al., 2022; Guo et al., 2025), and scaling this test-time computation continues to improve accuracy (Snell et al., 2025; Muennighoff et al., 2025). Current models often spend tens of thousands of tokens on a single question (Merrill et al., 2026; Motwani et al., 2026). At these lengths, attention during decoding is bound by memory reads of the KV cache (Shazeer, 2019; Pope et al., 2022). Each step loads the KV cache of every preceding token, so the cumulative KV traffic of a trace grows quadratically with its length. Query-aware sparse attention (Tang et al., 2024; Xiao et al., 2024; Chen et al., 2025) reduces this cost by keeping the full cache in memory but attending to only a selected subset at each step. Because these methods apply to pretrained models at inference time, they are a natural candidate for making long-form reasoning affordable.

However, attending to only a subset of the KV cache comes at a cost. It perturbs the attention outputs, the KV states that later layers compute from them, the cache at all future steps, and ultimately the next-token logits (Sun et al., 2026; Yang et al., 2025b). Generation under sparse attention therefore drifts from the trajectory that full attention would produce. This drift is especially pronounced in reasoning, where deviations compound over long generations (Sun et al., 2026). In practice, sparse decoding produces substantially longer generations and lower accuracy than dense decoding (Gao et al., 2026; Hu et al., 2026). This undermines the very motivation for sparsity: lower accuracy erodes its value, and the additional tokens can offset the per-step savings it is meant to deliver.

To understand this drift, we ask at which decoding steps sparse and dense decoding diverge. Comparing sparse and dense decoding on identical prefixes, we observe that the discrepancy between their next-token distributions is concentrated at steps where the dense distribution has high entropy (Sec 3.1). Larger per-step divergence, however, need not translate into worse final answers. To determine whether these divergences affect final performance, we decode with full attention but switch to sparse attention only at steps whose dense entropy is high, and compare against switching at an equal fraction of randomly chosen steps (Sec 3.2). Compared to randomly selected tokens at the

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.

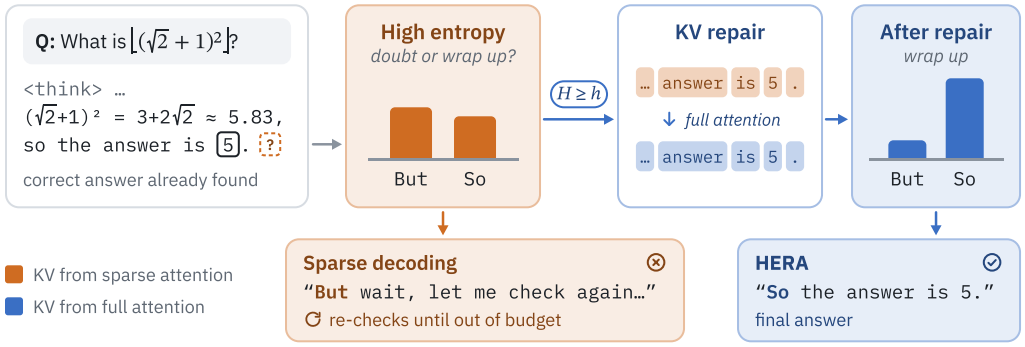


Figure 1: **Sparse decoding errors become most consequential at high-entropy steps.** HERA identifies these important steps using sparse next-token entropy, repairs the accumulated KV states with full attention, and samples the next token from the resulting dense distribution.

same rate, applying sparse attention at high-entropy tokens (about 15% of all tokens) lowers accuracy by 10.8 points instead of 1.4 and lengthens generations from 5.0k to 6.4k tokens instead of 5.3k tokens on Qwen3-8B and MATH-500 (Table 8). High-entropy steps are thus where sparse attention does most damage.

At each step, a sparse prediction differs from its dense counterpart due to two sources of error. *Local error* arises from attending to only a subset of past tokens at the current step. *Inherited error* arises from the cache itself, whose KV states have already been perturbed by earlier sparse decoding steps. At the same high-entropy intervention steps, we isolate each source by sparsifying either only the current step's attention or only the KV history, and find that inherited error accounts for most of the degradation at high-entropy steps (Sec 3.3). In Qwen3-8B on MATH-500, local error alone lowers accuracy by 1.2 points, whereas inherited error alone lowers it by 8.0 points. This suggests that restoring full attention for the current prediction alone is insufficient, as most of the error already resides in the cache.

Together, these findings suggest that dense computation should not be allocated uniformly during sparse decoding, as it is most useful when accumulated KV errors become the most detrimental. We propose HERA, a **H**igh-**E**ntropy-triggered KV **R**ep**A**ir method for decoding with sparse attention. As shown in Figure 1, at each step, HERA computes the next-token distribution using sparse attention and checks its entropy. When entropy exceeds a threshold, HERA recomputes the KV states of tokens generated since the previous repair using dense attention, updates the cache, and samples the next token from the resulting dense distribution. HERA uses sparse entropy as a proxy for the dense entropy used in our analysis, avoiding an additional dense forward pass to decide whether to repair. Section 4.1 shows that this proxy is effective, with an AUROC of 0.94–0.97 across thresholds.

We evaluate HERA on three reasoning models across four benchmarks spanning mathematics, scientific question answering, and code generation. HERA consistently improves accuracy and reduces generation length over the sparse-decoding baseline Quest (Tang et al., 2024), with accuracy gains of up to 18.8 percentage points and token reductions of up to 49.1%. Compared with the periodic KV-repair baseline ReSA (Sun et al., 2026), HERA generally achieves higher accuracy with fewer generated tokens and fewer repair calls.

Our contributions are as follows:

- We identify that sparse-attention errors are most detrimental at steps where the full-attention next token entropy is high. Only applying sparse attention at such steps severely degrades performance.
- We isolate *local* and *inherited* error at these steps and demonstrate that inherited error, the error from KV cache from past sparse decoding steps, accounts for most of the degradation.
- We propose HERA, a method that uses sparse next-token entropy to trigger KV repair: a full-attention pass that recomputes the past KV states. At comparable repair frequencies, HERA generally achieves higher accuracy with shorter generations than periodic repair.

2 Related Work

Cache drift under sparse decoding. Several works observe that sparse decoding accumulates error in the KV cache and degrades long generations (Yang et al., 2025b; Sun et al., 2026; Yang et al., 2026). Because attention outputs determine the keys and values of subsequent layers, the resulting distributional shift propagates across layers and into subsequent steps (Willette et al., 2025). In reasoning models, this drift also lengthens generation, potentially offsetting the per-step savings (Gao et al., 2026; Hu et al., 2026). Existing methods mitigate these errors in different ways. ReSA (Sun et al., 2026) periodically refreshes the accumulated KV cache with dense re-encoding. RetroAttention (Choi et al., 2026) retrospectively supplements past attention outputs with KV pages selected at later steps. HERA instead identifies when accumulated KV errors are most consequential and triggers dense history repair only at these high-entropy steps.

High-entropy tokens and generation. Prior work has explored the importance of high-entropy tokens. Wang et al. (2025a) show that high entropy tokens are critical in reinforcement learning and that restricting policy updates to a minority of high-entropy tokens preserves or improves reasoning performance. Chen et al. (2026) route high uncertainty tokens to a larger model in a speculative decoding setup. Furthermore, several works use predictive entropy to choose where to sample a reasoning branch (Xu et al., 2025; Wang et al., 2025b; Zhou et al., 2026; Scalena et al., 2025). Kai et al. (2026) use entropy as a heuristic for KV cache compression, and DiffAdapt (Liu et al., 2026) uses reasoning-trace entropy to choose an inference strategy. These results signal the importance of high entropy tokens in large language model decode.

3 Where Sparse Attention Breaks Reasoning

In this section, we examine at which decoding steps sparse predictions differ most from dense ones (Sec. 3.1). Then we test whether sparse predictions at those steps are what degrades reasoning (Sec. 3.2). Finally, we isolate the effects of local and inherited error at those critical steps (Sec. 3.3).

3.1 Where Sparse Predictions Depart from Dense Decoding

Motivation. To understand why sparse attention lengthens reasoning traces and lowers accuracy, we analyze at which steps along a trace sparse predictions depart from dense ones. Instead of comparing freely decoded sparse and dense trajectories, we fix a dense trajectory and create a new branch at each decode step, in which we regenerate the next token with sparse attention. In the sparse branch, each decoding step therefore sees the same prefix as its dense counterpart, but only attends to part of it.

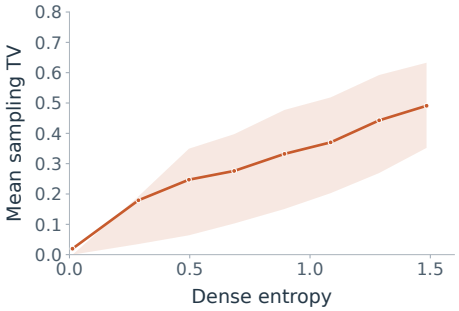


Figure 2: Mean TV between sparse and dense sampling distributions along fixed dense trajectories, plotted against dense entropy H_t. Shading indicates the interquartile range. **The sparse predicted distribution differs from the dense counterpart the most when entropy is high.**

Setup. Dense trajectories are generated by Qwen3-8B on MATH-500. Both branches encode the prompt with full attention. Over the fixed generated tokens, the dense branch keeps the original dense KV cache, while the sparse branch recomputes the KV cache with Quest and attends to it sparsely. Let p_t^D and p_t^S denote the next-token distributions of the dense and sparse branches at temperature 0.6 before top-p truncation, and let $H_t = -\sum_v p_t^D(v) \log p_t^D(v)$ denote the entropy of the dense branch at decode step t, over the vocabulary v. Applying top-p truncation and renormalization to each distribution yields the actual sampling distributions q_t^D and q_t^S. We measure their discrepancy by total variation distance, $\mathrm{TV}(q_t^D, q_t^S) = \frac{1}{2}\sum_v |q_t^D(v) - q_t^S(v)|$. TV is the fraction of probability mass on which the sparse and dense distributions disagree. It ranges from 0 when the two distributions are identical to 1 when their supports are disjoint.

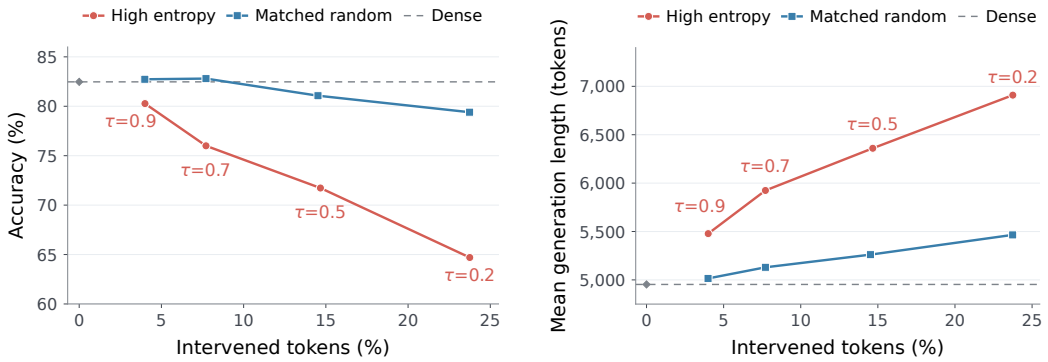


Figure 3: **Intervening at high-entropy steps lowers accuracy (left) and lengthens generations (right) far more than intervening at random steps at the same rate.** Each point on the high-entropy curve corresponds to one threshold τ, and the random policy is matched to its intervention rate. The dashed line is dense decoding. Exact numbers are reported in Table 8 (Appendix A.6).

Findings. Figure 2 shows that mean sampling TV increases with dense next-token entropy. Even with the same prefix, sparse attention induces larger changes in the sampling distribution at positions where dense decoding is more uncertain. This suggests that dense entropy is informative about local sensitivity to sparse attention. **Sparse predictions deviate from the dense counterparts most at high-entropy steps.** Conversely, at low-entropy positions the two distributions stay much closer.

3.2 Sparse Attention at High-Entropy Steps Degrades Reasoning

Motivation. Sec. 3.1 has shown that sparse and dense predictions diverge most at high-entropy positions. However, whether this discrepancy contributes to the degradation of reasoning quality is a separate question. We therefore test whether perturbing with sparsity at high-entropy steps is more responsible for the degradation than perturbing it at randomly selected steps.

Setup. We use Qwen3-8B on MATH-500 with Quest (Tang et al., 2024) at a token budget of 448 and a 12,288-token generation limit. We decode with full attention and switch to sparse attention only at selected steps to measure their sensitivity. We call the switch to sparse attention an *intervention*. At an intervened step t, we want the prediction that sparse decoding would have produced for the same prefix. To obtain it, we recompute the KV states of the tokens generated so far using sparse attention, let step t attend sparsely to these recomputed states, and sample x_t from the resulting distribution. The prediction therefore carries both local and inherited error. We then discard the recomputed states and continue with full attention from the original dense cache, so the only thing an intervention changes for later steps is the token x_t. The prompt is always encoded with full attention. We select the intervened steps in two ways. Let H_t be the entropy of the full-attention next-token distribution at step t. The high-entropy policy intervenes whenever $H_t \geq \tau$, and the random policy intervenes at randomly chosen steps with the same total number of intervened steps.

Findings. As shown in Figure 3, the difference is substantial between high-entropy and random policies. When intervening at steps with $H_t \geq 0.5$ (14.7% of all steps), accuracy drops from 82.5% to 71.7%, the average generation length grows from 5.0k to 6.4k tokens, and the fraction of responses that reach the generation limit rises from 8.1% to 22.5%. In contrast, random interventions at a matched rate leave all three nearly unchanged (81.1%, 5.3k tokens, and 10.3%). We report detailed results in Table 8.

The gap holds at every threshold and widens as τ is lowered. With a lower threshold, the high-entropy policy covers more steps (4.0% to 23.8% of the trace), and its accuracy falls steadily from 80.3% to 64.7% while generation length grows from 5.5k to 6.9k tokens. Random interventions at the same rates stay close to dense decoding throughout, losing at most 3.1 points of accuracy even when applied to 23.7% of steps. We observe the same qualitative sensitivity on GPQA-Diamond, where high-entropy interventions substantially lengthen generations while approximately rate-matched random interventions have little effect (Appendix A.10).

	Dense current		Sparse current	
History	Acc. (%)	Tokens	Acc. (%)	Tokens
Dense	82.5	4,953	81.3 (−1.2)	5,161
Sparse	74.5 (−8.0)	6,168	71.7 (−10.8)	6,360

Table 1: **Inherited error accounts for most of the degradation at high-entropy steps.** Results on Qwen3-8B and MATH-500 with interventions at $H_t \geq 0.5$. Sparse historical KV computation introduces inherited error, while sparse current-step attention introduces local error. Gray values show accuracy changes relative to dense decoding.

These results show that **high-entropy decisions are substantially more sensitive to sparse attention**. A comparable amount of sparse computation produces very different downstream effects depending on where it is applied.

3.3 Tracing the degradation to sparse KV history

Motivation. Sparse decoding perturbs the prediction at every step through both *local errors* that come from attending to only part of the past tokens, and *inherited errors* that come from the fact that past KV cache is computed by sparse attention. To understand how these perturbations turn into performance degradation, we isolate the two types of error at high-entropy steps ($H_t \geq 0.5$).

Setup. To isolate inherited error, we recompute the generated prefix's KV cache with sparse attention at intervention steps, and make the model attend to all past tokens. To isolate local error, we retain the dense KV states, and at intervention steps, the model sparsely attends to past KV states. Both interventions follow the protocol of Sec. 3.2: after sampling, the altered states are discarded and decoding continues from the dense cache. All non-intervened steps use full attention. Together with dense decoding and the combined intervention, these yield the four configurations in Table 1.

Findings. Table 1 shows that introducing inherited error alone lowers accuracy by 8.0 points (82.5% to 74.5%) and lengthens generations from 5.0k to 6.2k tokens. Local error alone costs only 1.2 points (81.3%) and leaves generation length almost unchanged (5.2k tokens).

This demonstrates that **most of the degradation comes from inherited error**. Using dense attention only at the current step is therefore not enough, as it recovers only 2.8 of the 10.8 lost points (71.7% to 74.5%), because the errors it would need to fix are already in the cache.

4 HERA: Entropy-triggered KV Cache Repair

Section 3 demonstrates that high-entropy steps are particularly sensitive to sparse attention. At these steps, most of the degradation comes from inherited errors which persist even when the current prediction uses full attention. These findings motivate HERA, which repairs the KV history before high-entropy predictions and samples from the resulting dense distribution, rather than allocating the dense recomputation periodically.

The remaining challenge is to identify these steps during sparse decoding.

Our analysis in Section 3 identifies high dense entropy as the relevant signal, but at deployment time we do not want to compute dense attention just to obtain that signal. We therefore ask whether the sparse entropy, which is already available, can serve as a proxy.

4.1 Sparse Entropy Is a Faithful Proxy for Dense Uncertainty

We use the paired dense and sparse predictions from the fixed-prefix replay in Sec. 3.1. Let H_t^D and H_t^S denote their next-token entropies, computed at temperature 0.6 before top-p truncation. For each dense-entropy threshold τ, we label steps with $H_t^D \geq \tau$ as high-entropy steps and evaluate how well H_t^S distinguishes them from the remaining steps using the area under the receiver operating characteristic curve (AUROC). We vary τ from 0.1 to 1.0.

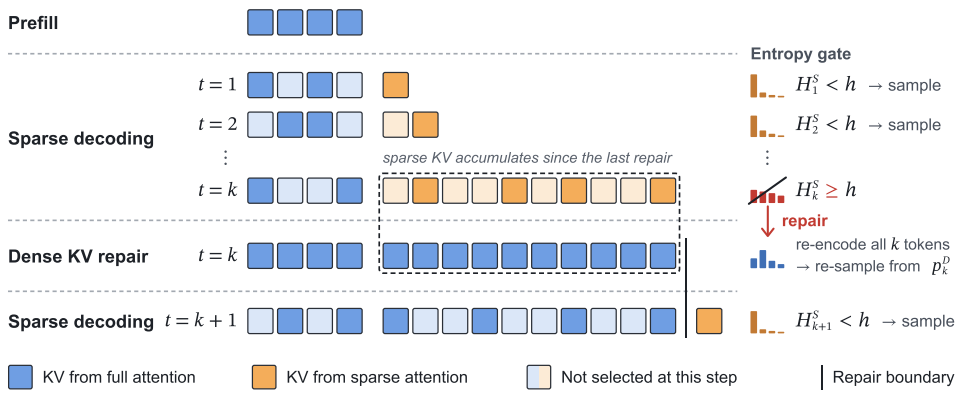


Figure 5: **Token-level workflow of HERA.** HERA decodes with sparse attention by default. When sparse next-token entropy exceeds the threshold, it re-encodes the tokens generated since the previous repair boundary with full attention, replaces their KV states, and samples from the resulting dense next-token distribution before resuming sparse decoding.

Figure 4 shows that sparse entropy consistently identifies decoding steps with high dense entropy, with AUROCs of approximately 0.94–0.97 across the tested thresholds. We therefore use sparse next-token entropy in HERA to decide when to repair the KV cache.

4.2 High-entropy Triggered KV Cache Repair

Figure 5 shows the workflow of HERA. It starts with a standard dense prefill over the prompt. During generation, each decoding step first follows the sparse attention backend to produce a next-token distribution. Let H_t^S denote the entropy of the sparse next-token distribution at step t. If $H_t^S < h$, HERA samples the next token from this sparse distribution and proceeds to the next sparse decoding step. As generation continues in this way, the KV states of newly generated tokens are computed under sparse attention and accumulate in the cache.

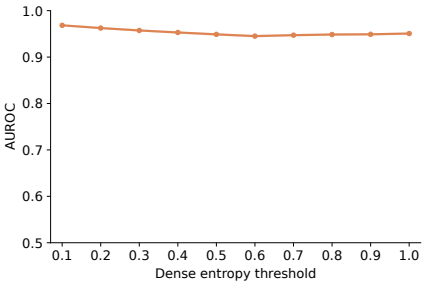


Figure 4: AUROC of sparse entropy for identifying steps with dense entropy $\geq \tau$, across different thresholds τ on Qwen3-8B and MATH-500. **Sparse entropy consistently identifies decoding steps with high dense entropy.**

When $H_t^S \geq h$, HERA pauses sparse decoding and repairs the KV states accumulated since the previous repair. Specifically, it replays those generated tokens with full causal attention, using the already repaired prefix KV states as context, and replaces sparsely computed KV states with the recomputed dense ones. The replay also yields the dense next-token distribution at step t, from which HERA samples the next token. Sparse decoding then resumes from the repaired cache.

To avoid triggering multiple repairs in close succession, we require at least m generated tokens between consecutive repairs. We use $m = 8$ in our experiments.

5 Experiments

5.1 Experimental Setup

Models and benchmarks. We evaluate three reasoning models: Qwen3-8B and Qwen3-4B in thinking mode (Yang et al., 2025a), and DeepSeek-R1-Distill-Llama-8B (Guo et al., 2025). We use four benchmarks: MATH-500 (500 mathematical problems; Lightman et al. (2023)), AIME 2025 (30 competition problems; Art of Problem Solving (2025)), GPQA-Diamond (198 multiple-choice science questions; Rein et al. (2023)), and LiveCodeBench v6 (175 code generation problems; Jain et al. (2024)).

Decoding settings. All methods use dense prefill, temperature 0.6, and top-p 0.95. For controlled comparison, Quest, ReSA-k, and HERA use the same Quest-style sparse-attention backend with a page size of 64 tokens and per-query-head page selection. Rather than using ReSA's GQA-grouped implementation, we instantiate its periodic rectification policy on this common backend, so that ReSA and HERA differ primarily in when dense KV repair is triggered. We set the attention-read budget of each benchmark to roughly 10% of the dense model's mean generation length, which gives 448, 2,048, 1,024, and 1,024 tokens for MATH-500, AIME 2025, GPQA-Diamond, and LiveCodeBench, respectively. The maximum generation length is 32,768 tokens, except for Qwen3-8B on MATH-500, where we use 12,288 tokens.

Compared Methods. **(1) Dense** uses full attention throughout decoding. **(2) Quest** selects KV pages according to query-dependent scores computed from page-level key extrema (Tang et al., 2024). **(3) ReSA-k** ($k \in \{16, 32\}$) applies periodic KV rectification every k generated-token forward steps, using the same sparse backend and attention budget as Quest (Tang et al., 2024; Sun et al., 2026). **(4) HERA** uses the same sparse backend and attention budget as ReSA. We treat the entropy threshold h as a compute-control parameter rather than a performance-tuned hyperparameter. For comparison with ReSA-k, we estimate candidate thresholds from the entropy distribution of Quest baseline trajectories and choose values that approximately match its repair frequency. This procedure uses neither correctness labels nor HERA accuracy (see Appendix A.5).

Metrics. We report accuracy (Acc.), mean generated tokens (Tokens), mean dense repair calls (Repairs), and estimated decoding KV-memory access (KV Access) which follows the analytical model of Sun et al. (2026) (detailed in Appendix A.2). Truncation rates are reported in Appendix A.9.

5.2 Main Results

HERA mitigates the degradation from sparse decoding. Across three models and all four benchmarks (Tables 2, 3 and 4), HERA improves or matches Quest in accuracy while generating fewer tokens. The improvement is most pronounced on Qwen3-8B with MATH-500, where HERA ($h = 0.5$) increases accuracy from 59.3% to 78.1% while reducing the mean generation length from 7,383 to 5,484 tokens. Similar improvements hold across other tasks, showing that selectively repairing the accumulated KV errors substantially recovers the degradation caused by sparse decoding. The shorter generations can also offset the added repair accesses: on Qwen3-4B/MATH-500, HERA ($h = 0.5$) has lower estimated KV-memory access than Quest (0.642× vs. 1.230× Dense).

Entropy-triggered repair makes more effective use of dense correction. At approximately matched repair frequencies, HERA generally achieves higher accuracy with shorter generations than ReSA. On Qwen3-4B/MATH-500, HERA ($h = 0.5$) further reduces estimated decoding KV-memory access by 27.7% relative to ReSA-16 while improving accuracy by 3.0 points. HERA ($h = 0.9$) reduces access by 20.2% while improving accuracy by 2.0 points. Thus, the gains are not explained by more frequent dense correction, but by *when* it is applied. Targeting high-entropy steps makes better use of dense repair.

A **matched-random control** further supports our conclusion: at nearly identical repair frequencies, HERA improves accuracy by 3.2–3.8 points while generating 16.0–16.6% fewer tokens than random repair (Appendix A.3).

5.3 Impact of KV history repair

At high-entropy steps, **HERA** removes inherited errors by repairing KV states with full attention, and removes local error by attending to all past tokens. We ablate the effect of repairing both types of errors: we compare HERA with the following two variants:

- HERA with inherited errors: At high-entropy steps, we do not repair the past KV but make the next token attend to all past tokens — densely attending to a sparsely computed KV.
- HERA with local errors: At high-entropy steps, we do repair the past KV but make the next token only attend to part of the dense KV — sparsely attending to a densely computed KV.

Table 2: Qwen3-8B results across four reasoning benchmarks. Parenthesized values beside HERA denote the entropy threshold h. Tokens and Repairs are means per trajectory, and the parenthesized percentages in HERA rows indicate changes relative to the ReSA configuration immediately above. Accuracy is a percentage; KV Access is normalized by Dense within each model–benchmark setting. Dashed lines separate the dense baseline from sparse methods.

MATH-500 (500 q.)					**AIME 2025** (30 q.)				
Method	Acc.↑	Tokens↓	Repairs↓	KV Access↓	Method	Acc.↑	Tokens↓	Repairs↓	KV Access↓
Dense	82.5	4,953	0.0	1.000	Dense	71.1	17,501	0.0	1.000
Quest	59.3	7,383	0.0	0.772	Quest	66.7	19,679	0.0	0.822
ReSA-16	72.5	5,920	369.4	0.689	ReSA-16	65.6	18,080	1,129.4	0.813
HERA (0.5)	**78.1**	5,484 (−7.4%)	318.8 (−13.7%)	0.625 (−9.3%)	**HERA** (0.5)	67.8	17,523 (−3.1%)	995.4 (−11.9%)	0.772 (−5.0%)
ReSA-32	70.4	6,257	194.9	0.689	ReSA-32	64.4	18,796	586.7	0.815
HERA (0.9)	73.3	5,864 (−6.3%)	161.1 (−17.4%)	0.640 (−7.1%)	**HERA** (0.9)	**68.9**	17,741 (−5.6%)	485.4 (−17.3%)	0.756 (−7.2%)
GPQA-Diamond (198 q.)					**LiveCodeBench v6** (175 q.)				
Method	Acc.↑	Tokens↓	Repairs↓	KV Access↓	Method	Acc.↑	Tokens↓	Repairs↓	KV Access↓
Dense	62.6	9,776	0.0	1.000	Dense	49.4	14,525	0.0	1.000
Quest	54.3	15,963	0.0	1.154	Quest	39.4	22,040	0.0	0.691
ReSA-16	60.4	11,008	687.5	0.822	ReSA-16	**49.7**	17,157	1,071.7	0.605
HERA (0.5)	**61.9**	10,351 (−6.0%)	774.7 (+12.7%)	0.770 (−6.3%)	**HERA** (0.9)	**49.7**	16,870 (−1.7%)	762.8 (−28.8%)	0.566 (−6.4%)
ReSA-32	60.4	11,291	352.3	0.804	ReSA-32	48.9	18,166	567.0	0.604
HERA (1.1)	61.1	11,226 (−0.6%)	311.6 (−11.6%)	0.795 (−1.1%)	**HERA** (1.05)	**49.7**	17,562 (−3.3%)	572.5 (+1.0%)	0.576 (−4.6%)

Table 3: Qwen3-4B results across four reasoning benchmarks. Notation follows Table 2.

MATH-500 (500 q.)					**AIME 2025** (30 q.)				
Method	Acc.↑	Tokens↓	Repairs↓	KV Access↓	Method	Acc.↑	Tokens↓	Repairs↓	KV Access↓
Dense	86.2	5,213	0.0	1.000	Dense	60.0	17,630	0.0	1.000
Quest	67.6	13,025	0.0	1.230	Quest	63.3	18,462	0.0	0.728
ReSA-16	82.6	8,225	513.5	0.888	ReSA-16	70.0	17,812	1,112.5	0.764
HERA (0.5)	**85.6**	6,624 (−19.5%)	408.6 (−20.4%)	0.642 (−27.7%)	**HERA** (0.5)	**73.3**	18,296 (+2.7%)	1,084.9 (−2.5%)	0.781 (+2.2%)
ReSA-32	80.6	9,121	284.5	0.914	ReSA-32	60.0	17,743	553.9	0.729
HERA (0.9)	82.6	7,663 (−16.0%)	241.2 (−15.2%)	0.729 (−20.2%)	**HERA** (0.9)	66.7	17,646 (−0.5%)	514.4 (−7.1%)	0.717 (−1.6%)
GPQA-Diamond (198 q.)					**LiveCodeBench v6** (175 q.)				
Method	Acc.↑	Tokens↓	Repairs↓	KV Access↓	Method	Acc.↑	Tokens↓	Repairs↓	KV Access↓
Dense	55.1	8,904	0.0	1.000	Dense	49.1	14,897	0.0	1.000
Quest	50.5	14,076	0.0	1.123	Quest	36.0	22,357	0.0	0.674
ReSA-16	55.6	10,119	631.9	0.836	ReSA-16	48.6	17,648	1,102.4	0.603
HERA (0.7)	**56.1**	9,599 (−5.1%)	587.9 (−7.0%)	0.776 (−7.2%)	**HERA** (0.7)	**48.6**	16,260 (−7.9%)	1,008.7 (−8.5%)	0.537 (−10.9%)
ReSA-32	52.5	10,615	331.2	0.848	ReSA-32	48.0	18,259	569.9	0.584
HERA (1.1)	55.1	10,487 (−1.2%)	306.0 (−7.6%)	0.828 (−2.4%)	**HERA** (1.1)	47.4	17,694 (−3.1%)	487.4 (−14.5%)	0.552 (−5.5%)

The setup in this section differs from the results in Sec. 3.3. There, we use standard dense decoding at non-intervened steps, whereas in this section, we use sparse decoding (Quest).

Table 5 shows that both errors contribute to the performance degradation, but inherited error is more significant. HERA with inherited errors reduces accuracy from 81.0% to 74.0%, whereas HERA with local errors reduces it to 77.5%. The former also produces substantially longer generations (6,293 vs. 5,104 tokens) and a higher truncation rate (23.0% vs. 15.0%) than the latter. These results are consistent with the inherited-error analysis in Sec. 3.3. Repair also materially changes the resulting prediction: in a diagnostic analysis, 39–53% of repair events change the top-1 token, with mean distributional TV of 0.33–0.42 (Appendix A.4).

5.4 How Does HERA Change Reasoning Trajectories?

We analyze Qwen3-4B trajectories on MATH-500 to understand where HERA changes the outcome. We compare HERA ($h = 0.5$) with Quest and ReSA-16 on the same 500 questions. We call a baseline-incorrect/HERA-correct pair a *rescue*, and the reverse a *regression*.

Table 4: DeepSeek-R1-Distill-Llama-8B results. Notation follows Table 2.

MATH-500 (500 q.)					**AIME 2025** (30 q.)				
Method	Acc.↑	Tokens↓	Repairs↓	KV Access↓	Method	Acc.↑	Tokens↓	Repairs↓	KV Access↓
Dense	83.0	4,356	0.0	1.000	Dense	33.3	12,875	0.0	1.000
Quest	75.9	8,155	0.0	0.836	Quest	31.7	16,547	0.0	1.091
ReSA-16	82.5	4,853	302.7	0.517	ReSA-16	38.3	13,352	834.0	0.911
HERA (0.7)	**83.7**	4,673 (−3.7%)	266.6 (−11.9%)	0.482 (−6.8%)	**HERA** (0.9)	**40.0**	13,326 (−0.2%)	631.9 (−24.2%)	0.894 (−1.9%)
ReSA-32	81.2	5,734	178.7	0.596	ReSA-32	**40.0**	16,041	500.7	1.097
HERA (1.3)	81.7	5,308 (−7.4%)	161.5 (−9.6%)	0.542 (−9.1%)	**HERA** (1.2)	35.0	13,766 (−14.2%)	448.4 (−10.4%)	0.911 (−17.0%)
GPQA-Diamond (198 q.)					**LiveCodeBench v6** (175 q.)				
Method	Acc.↑	Tokens↓	Repairs↓	KV Access↓	Method	Acc.↑	Tokens↓	Repairs↓	KV Access↓
Dense	40.7	9,440	0.0	1.000	Dense	34.6	12,275	0.0	1.000
Quest	40.7	11,908	0.0	0.726	Quest	35.4	19,227	0.0	0.799
ReSA-16	44.2	8,368	522.5	0.514	ReSA-16	34.3	11,487	717.4	0.485
HERA (0.8)	**44.4**	8,276 (−1.1%)	515.8 (−1.3%)	0.505 (−1.8%)	**HERA** (0.9)	**36.9**	11,439 (−0.4%)	730.3 (+1.8%)	0.478 (−1.4%)
ReSA-32	42.4	8,726	272.2	0.523	ReSA-32	35.1	11,614	362.4	0.466
HERA (1.3)	42.9	8,532 (−2.2%)	265.0 (−2.6%)	0.505 (−3.4%)	**HERA** (1.35)	35.4	11,681 (+0.6%)	370.5 (+2.2%)	0.466 (0.0%)

Table 5: Impact of KV history repair on the first 100 MATH-500 problems with Qwen3-8B, using a maximum generation length of 12,288 tokens. Results are averaged over two sampling seeds.

Method	Acc. (%)	Tokens	Trunc. (%)
Dense	84.5	4,435	6.5
Quest (Tang et al., 2024)	66.0	6,839	31.5
HERA with local errors	77.5	5,104	15.0
HERA with inherited errors	74.0	6,293	23.0
HERA ($h = 0.5$)	81.0	4,827	12.5

Most rescues involve truncated sparse trajectories. Relative to Quest, HERA rescues 90 questions with no scored regressions, and 88 of these 90 rescued Quest trajectories reach the generation limit. Relative to ReSA-16, HERA rescues 20 questions and regresses on five, with 16 of the 20 rescues involving truncated ReSA-16 trajectories. Thus, most observed rescues are associated with sparse trajectories that fail to terminate within the generation budget.

HERA can avoid prolonged rechecking after reaching a solution. For example, on MATH-500 problem 245 (Figure 6 in Appendix A.8), both Quest and HERA derive the correct remainder, $-13x + 3$, early in the trajectory. Quest subsequently repeats calculations and revisits previously verified steps until reaching the 32,768-token limit without completing its response. HERA reaches the same result, performs additional checks, and terminates after 4,841 tokens.

These trajectory-level observations do not imply that individual repair events directly correct semantic reasoning errors. HERA does not revise previously generated tokens, and some scored changes can also reflect answer formatting or extraction.

6 Conclusion

We study why sparse attention degrades long-form reasoning and find that its impact is concentrated at high-entropy decoding steps, where errors accumulated in the KV history account for most of the degradation. We propose HERA, which translates this diagnosis into an adaptive repair policy that concentrates dense computation at these consequential steps. It uses sparse next-token entropy to selectively trigger dense KV repair. Across models and reasoning benchmarks, HERA generally achieves higher accuracy and shorter generations than periodic repair at comparable repair frequencies.

AI USE STATEMENT

We used generative AI tools to polish the writing of this paper, including grammar, wording, and readability. We also used generative AI to draft some portions of the appendix, all of which were reviewed and revised by the authors. The research ideas, methodology, experiments, and results are the authors' own. We take full responsibility for the content of this work.

REPRODUCIBILITY STATEMENT

We describe HERA in Section 4.2 and provide the experimental setup in Section 5, including the models (Qwen3-4B, Qwen3-8B, and DeepSeek-R1-Distill-Llama-8B), benchmarks, sparse attention settings, and all hyperparameters. For each model and benchmark, the entropy threshold h is selected to match the repair budget of ReSA, and the minimum repair interval is fixed to $m = 8$.

REFERENCES

Art of Problem Solving. 2025 AIME I problems and solutions. https://artofproblemsolving.com/wiki/index.php/2025_AIME_I, 2025. Accessed: September 26, 2026.

Zhuokun Chen, Zeren Chen, Jiahao He, Lu Sheng, Mingkui Tan, Jianfei Cai, and Bohan Zhuang. R-stitch: Dynamic trajectory stitching for efficient reasoning, 2026. URL https://openreview.net/forum?id=h1t9kRSbLb.

Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, and Beidi Chen. MagicPIG: LSH sampling for efficient LLM generation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ALzTQUgW8a.

Seonghwan Choi, Beomseok Kang, Dongwon Jo, and jae-joon kim. Retrospective sparse attention for efficient long-context generation. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 150937–150960, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/f4daa773a5bb2d562a9204a7e2225a67-Paper-Conference.pdf.

Yizhao Gao, Shuming Guo, Shijie Cao, Yuqing Xia, Yu Cheng, Lei Wang, Lingxiao Ma, Yutao Sun, Tianzhu Ye, Li Dong, Hayden Kwok-Hay So, Yu Hua, Ting Cao, Fan Yang, and Mao Yang. Sparse attention adaptation for long reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=c5BOcHM6J8.

Daya Guo, Dejian Yang, Haowei Zhang, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645:633–638, 2025. doi: 10.1038/s41586-025-09422-z.

Junhao Hu, Fangze Li, Mingtao Xu, Feifan Meng, Shiju Zhao, Tiancheng Hu, Ting Peng, Anmin Liu, Wenrui Huang, Chenxu Liu, Ziyue Hua, and Tao Xie. Lil: Less is less when applying post-training sparse-attention algorithms in long-decode stage. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 1898–1912, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.91. URL https://aclanthology.org/2026.findings-acl.91/.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Jushi Kai, Zhuiri Xiao, Alexandra Birch, and Zhouhan Lin. Information-aware kv cache compression for long reasoning, 2026. URL https://arxiv.org/abs/2606.26875.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Xiang Liu, Xuming Hu, Xiaowen Chu, and Eunsol Choi. Diffadapt: Difficulty-adaptive reasoning for token-efficient LLM inference. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=644FH1vVIl.

Mike A Merrill, Alexander Glenn Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Di Lu, Orfeas Menis Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Kumar Guha, Gabriel H. S. Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Kwesi Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong Wang, Changzhi

Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Christopher Michael Rytting, Ryan Marten, Yixin Wang, Jenia Jitsev, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=a7Qa4CcHak.

Sumeet Ramesh Motwani, Daniel Nichols, Charles London, Peggy Li, Fabio Pizzati, Acer Blake, Hasan Abed Al Kader Hammoud, Tavish Malcolm McDonald, Akshat Naik, Alesia Ivanova, Vignesh Baskaran, Ivan Laptev, Ruben Glatt, Tal Ben-Nun, Philip Torr, Ameya Prabhu, Brian R. Bartoldson, Bhavya Kailkhura, and Christian Schroeder de Witt. Longcot: Benchmarking long-horizon chain-of-thought reasoning. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=47NnSXz3im.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candes, and Tatsunori Hashimoto. s1: Simple test-time scaling. In *Workshop on Reasoning and Planning for Large Language Models*, 2025. URL https://openreview.net/forum?id=LdH0vrgAHm.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference, 2022. URL https://arxiv.org/abs/2211.05102.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Daniel Scalena, Leonidas Zotos, Elisabetta Fersini, Malvina Nissim, and Ahmet Üstün. Eager: Entropy-aware generation for adaptive inference-time scaling. *arXiv preprint arXiv:2510.11170*, 2025.

Noam Shazeer. Fast transformer decoding: One write-head is all you need, 2019. URL https://arxiv.org/abs/1911.02150.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

Yutao Sun, Tianzhu Ye, Li Dong, Yuqing Xia, Jian Chen, Yizhao Gao, Shijie Cao, Jianyong Wang, and Furu Wei. Rectified sparse attention for efficient long-sequence generation. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 7023–7034, 2026.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. QUEST: Query-aware sparsity for efficient long-context LLM inference. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=KzACYw0MTV.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=yfcpdY4gMP.

Yizhou Wang, Lingzhi Zhang, Yue Bai, Mang Tik Chiu, Zhengmian Hu, Mingyuan Zhang, Qihua Dong, Yu Yin, Sohrab Amirghodsi, and Yun Fu. Cautious next token prediction. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 25685–25697, Vienna, Austria, July 2025b. Association for Computational Linguistics. ISBN 979-8-89176-256-5. URL https://aclanthology.org/2025.findings-acl.1318/.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_VjQlMeSB_J.

Jeffrey Willette, Heejun Lee, and Sung Ju Hwang. Delta attention: Fast and accurate sparse attention inference by delta correction. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=LZrRvYBqsJ.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. InfLLM: Training-free long-context extrapolation for LLMs with an efficient context memory. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=bTHFrqhASY.

Zicheng Xu, Guanchu Wang, Yu-Neng Chuang, Guangyao Zheng, Alex Szalay, Zirui Liu, and Vladimir Braverman. DTS: Enhancing large reasoning models via decoding tree sketching. In *NeurIPS 2025 Workshop on Efficient Reasoning*, 2025. URL https://openreview.net/forum?id=WqhaykkEed.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

Lijie Yang, Zhihao Zhang, Zhuofu Chen, Zikun Li, and Zhihao Jia. Tidaldecode: Fast and accurate llm decoding with position persistent sparse attention. In *International Conference on Learning Representations*, volume 2025, pp. 5823–5844, 2025b.

Lijie Yang, Zhihao Zhang, Arti Jain, Shijie Cao, Baihong Yuan, Yiwei Chen, Zhihao Jia, and Ravi Netravali. Less is more: Fast and accurate reasoning with cross-head unified sparse attention, 2026. URL https://openreview.net/forum?id=3iwDzfIk60.

Felix Zhou, Anay Mehrotra, and Quanquan C. Liu. Reasoning with sampling: Cutting at decision points, 2026. URL https://arxiv.org/abs/2605.30327.

A APPENDIX

A.1 LIMITATIONS

Wall-clock efficiency. We evaluate computational efficiency using an analytical KV-memory-access model rather than wall-clock latency. A fair latency comparison requires equally optimized, method-specific GPU kernels, which our current implementation does not provide. In particular, our ReSA (Sun et al., 2026) baseline instantiates periodic KV rectification on the same per-query-head Quest backend used by HERA, rather than using its original GQA-grouped implementation. This controlled design isolates the effect of repair scheduling, but does not constitute a systems-level comparison with the original ReSA (Sun et al., 2026) implementation.

Besides, *our primary contribution is to identify when sparse-attention errors become consequential and to develop an adaptive repair policy targeting these steps*. Developing optimized kernels and evaluating end-to-end latency is left for future work.

Sparse-attention backends. Our experiments instantiate sparse decoding with Quest (Tang et al., 2024). Although HERA only requires a sparse next-token distribution and the ability to repair accumulated KV states, we have not evaluated whether the same entropy-triggered repair behavior generalizes to other sparse-attention mechanisms.

A.2 ANALYTICAL MEMORY-ACCESS ESTIMATE

We estimate decoding KV-memory access using an analytical model adapted from ReSA (Sun et al., 2026). Our goal is to compare Quest, periodic repair, and HERA under a common sparse-attention implementation rather than reproduce implementation-specific kernel behavior. All sparse methods therefore use the same Quest-style backend with per-query-head page selection. The analysis below provides a common accounting framework for the resulting decoding trajectories and repair schedules.

Cost model. We express memory access in KV-token equivalents per KV head, omitting common factors from layers, head dimension, and element size. Let $g = H_q/H_{\mathrm{KV}}$ denote the number of query heads sharing each KV head; $g = 4$ for all models evaluated in this work. For decoding step t, let L_t denote the current KV-cache length, b the page size, and $B = kb$ the sparse-attention budget.

Our sparse backend performs page selection independently for each query head, whereas dense attention and dense rectification can reuse the KV states within a GQA group. For an ordinary decoding forward of a sparse method, we therefore estimate

$$C_t^{\mathrm{sparse}} = \begin{cases} L_t, & L_t \le B, \\ g\left(\frac{L_t}{b} + A_t\right), & L_t > B, \end{cases} \tag{1}$$

where the first case accounts for the backend's dense fallback on short contexts. For $L_t > B$, the backend selects k pages per query head and always retains the final page. The number of valid selected KV tokens per query head is therefore reconstructed as

$$A_t = (k-1)b + 1 + ((L_t - 1) \bmod b). \tag{2}$$

This quantity reconstructs selection cardinality from the page-selection rule rather than logged page indices. In particular, it does not measure overlap or reuse of selected pages across query heads.

Following ReSA's batched-rectification abstraction, we count each dense repair as one additional full-history KV access. For a trajectory with ordinary decoding forwards $\mathcal{D}$ and repair positions $\mathcal{R}$,

$$C_{\mathrm{mem}} = \sum_{t \in \mathcal{D}} C_t^{\mathrm{sparse}} + \sum_{r \in \mathcal{R}} L_r. \tag{3}$$

In our recorded runs, $|\mathcal{D}| = G - 1$ for a trajectory with G generated tokens because the first token is sampled from the prefill output. For HERA, the sparse forward used to compute the triggering entropy is included even when it is immediately followed by a repair.

For dense decoding, we use

$$C_{\mathrm{mem}}^{\mathrm{Dense}} = \sum_{t \in \mathcal{D}} L_t. \tag{4}$$

Table 6: **Entropy-triggered repair outperforms random repair at matched repair frequencies.** Results on Qwen3-4B and MATH-500 over all 500 problems. Repair frequency denotes repair calls per generated token.

Method	Acc. (%)↑	Tokens↓	Repairs↓	Freq. (%)
Random, matched to $h = 0.5$	82.4	7,884	485.4	6.158
HERA ($h = 0.5$)	85.6	6,624	408.6	6.170
Random, matched to $h = 0.9$	78.8	9,190	289.0	3.145
HERA ($h = 0.9$)	82.6	7,663	241.2	3.149

We exclude the common dense prefill from all methods while including the prompt KV states in L_t during subsequent decoding. We aggregate the estimated access over all trajectories and normalize by the corresponding aggregate Dense cost.

This is an analytical access estimate rather than a measurement of physical memory traffic or latency.

A.3 Matched-Random Repair Control

To isolate the effect of *where* repair is applied, we compare HERA against a random-trigger repair policy on Qwen3-4B and MATH-500. For each HERA threshold, the random policy is calibrated to the same trigger frequency, while using the same minimum repair interval and dense-replay implementation. Both policies reuse the logits produced by dense replay for next-token sampling.

Table 6 reports results over all 500 problems with one sampling seed. The realized repair frequencies are nearly identical: 6.158% versus 6.170% for the $h = 0.5$ pair, and 3.145% versus 3.149% for the $h = 0.9$ pair. Despite this, HERA consistently achieves higher accuracy and shorter generations. At $h = 0.5$, HERA improves accuracy by 3.2 percentage points and reduces generation length by 16.0%; at $h = 0.9$, it improves accuracy by 3.8 points and reduces generation length by 16.6%. Because the resulting trajectories are shorter, HERA also requires fewer total repairs. These results provide direct evidence that the gains come from *where* dense repair is applied rather than from repairing more frequently.

A.4 What Changes When HERA Repairs?

Section 3 motivates repair at high-entropy positions, but it does not show how repair changes the next-token distribution. Moreover, Section 3 analyzes fixed dense trajectories, whereas HERA repairs its own trajectories, in which earlier repairs have already restored part of the cache. In this section, we therefore analyze how HERA repairs change the next-token distribution along HERA trajectories.

Setup and measurement. We analyze HERA trajectories generated by Qwen3-8B on GPQA-Diamond with thresholds $h = 0.5$ and $h = 0.9$, using the first 118 questions. We use a Quest budget of 448 tokens and a 12,288-token generation limit, and HERA enforces a minimum interval of 8 tokens between consecutive repairs. Sampling uses temperature 0.6 and top-p 0.95. This setting differs from the main GPQA-Diamond experiment (Section 5.1), which uses a 1,024-token budget and a 32,768-token limit on all 198 questions.

At a repair-triggering position t, let p_t^- be the sparse next-token distribution before repair, and let p_t^+ be the dense distribution obtained after recomputing the KV states of the tokens generated since the previous repair. Both are full-vocabulary distributions at temperature 0.6 before top-p truncation. We compute

$$H_t^{\pm} = -\sum_{v\in\mathcal{V}} p_t^{\pm}(v)\log p_t^{\pm}(v), \qquad \Delta H_t = H_t^- - H_t^+, \qquad \mathrm{TV}_t = \tfrac{1}{2}\sum_{v\in\mathcal{V}} \left|p_t^-(v) - p_t^+(v)\right|, \quad (5)$$

where a positive ΔH_t means that repair lowers the next-token entropy, and TV_t measures how much repair changes the distribution.

Table 7: **Repair reduces entropy on average and frequently changes the leading candidate.** Effect of HERA repairs for Qwen3-8B on 118 GPQA-Diamond questions. *Repairs*: total number of repairs. *Mean* $H_t^- \to H_t^+$: mean next-token entropy before and after repair. *All-step* H: mean next-token entropy over all HERA decoding steps. *Decreased*: percentage of repairs after which entropy decreased. *Below* h: percentage of repairs whose post-repair entropy falls below the threshold h. *Argmax changed*: percentage of repairs that change the top-1 token. *Mean TV*: mean total variation distance between the full-vocabulary distributions before and after repair.

h	Repairs	Mean $H_t^- \to H_t^+$	All-step H	Decreased	Below h	Argmax changed	Mean TV
0.5	88,768	$0.8396 \to 0.6686$	0.2310	66.5%	32.7%	39.3%	0.3266
0.9	52,229	$1.1586 \to 0.8587$	0.2285	75.5%	51.0%	53.0%	0.4170

Repair lowers entropy on average but does not remove it. Table 7 summarizes the effect of repair. Entropy decreases in 66.5% and 75.5% of repairs for $h = 0.5$ and $h = 0.9$, respectively (mean $\Delta H_t = 0.1710$ and 0.2999), and 32.7% and 51.0% of repairs end below the threshold, i.e., the sparse entropy that triggered them was at least partly induced by sparse error.

Repair often changes the sampled continuation. The top-1 token changes in 39.3% and 53.0% of repairs, with mean TV of 0.3266 and 0.4170. Because HERA samples from p_t^+, these changes directly alter the trajectory.

Repaired positions remain uncertain. Across all HERA decoding steps, mean entropy is 0.2310 ($h = 0.5$) and 0.2285 ($h = 0.9$), whereas post-repair entropy at triggered positions is 0.6686 and 0.8587; 84.8% and 90.7% of repaired positions remain above the all-step mean. Thus HERA spends dense computation on intrinsically uncertain steps rather than only removing sparsity-induced uncertainty, consistent with Section 3. Two caveats apply: because repairs are selected for high pre-repair entropy, part of the observed decrease reflects regression to the mean; and without a fully dense reference at every prefix, these results do not show that sparse entropy exactly recovers dense-model uncertainty.

A.5 How does HERA choose h?

For configurations using frequency-based calibration, we collect frozen Quest trajectories on a separate unlabeled calibration set. For each candidate threshold h, we replay the HERA gate ($H_t^S \geq h$) with the same minimum repair interval $m = 8$ used at inference time, and count the resulting repair events. We select h to approximately match the repair frequency of ReSA-16 or ReSA-32 on trajectories of the same lengths. The selected threshold is then fixed for evaluation. This procedure uses neither correctness labels nor HERA outputs.

A.6 Full Results of the Intervention Study

Table 8 reports the full results of the intervention study in Sec. 3.2, including accuracy, generation length, and truncation rate at each entropy threshold. Across thresholds, high-entropy interventions consistently cause substantially greater degradation than approximately rate-matched random interventions.

A.7 Which Tokens Trigger Repair?

We also examine which tokens precede repair, i.e., the last generated token before each repair-triggering position. Frequent tokens appear often at repair positions simply because they are frequent, so raw counts alone are uninformative. We therefore also report a relative frequency. For a token v, let $f_r(v)$ be the fraction of repair-triggering positions preceded by v, and let $f_n(v)$ be the corresponding fraction among non-repair positions. We define

$$F(v) = \frac{f_r(v)}{f_n(v)}, \tag{6}$$

Table 8: **Sparse attention is substantially more harmful at high-entropy steps than at randomly selected steps with similar intervention rates.** Results on MATH-500 with Qwen3-8B and Quest (token budget 448). Sparse attention is applied only at intervened steps, while all other steps use full attention. High-entropy interventions are selected by $H_t \geq \tau$; random interventions are approximately matched to their intervention rates. Results are averaged over three seeds with a 12,288-token generation limit.

	High entropy				Approximately rate-matched random			
τ	Rate (%)	Acc. (%)↑	Tokens↓	Trunc. (%)↓	Rate (%)	Acc. (%)↑	Tokens↓	Trunc. (%)↓
Dense	0.00	82.47	4,953	8.13		–		
0.2	23.75	64.70	6,909	29.50	23.74	79.40	5,465	12.50
0.5	14.67	71.73	6,360	22.47	14.52	81.07	5,261	10.33
0.7	7.71	76.00	5,924	17.10	7.71	82.80	5,130	9.30
0.9	4.00	80.27	5,478	12.13	3.99	82.73	5,015	8.87

Table 9: **Most frequent repair-preceding tokens.** Tokens are sorted by their combined repair count over both thresholds. F is the relative frequency defined in Appendix A.7. Repairs are most often preceded by punctuation (e.g., ".", ",", "?"), conjunctions (e.g., "So", "but"), and function words (e.g., "the", "is", "to").

Preceding token	$h = 0.5$ count	F	$h = 0.9$ count	F
`"."`	9,835	3.28×	7,169	3.43×
`","`	8,863	2.89×	7,070	3.19×
`" the"`	4,991	2.19×	3,831	2.09×
`" is"`	2,774	1.96×	1,677	1.63×
`" \n\n"`	2,216	6.99×	1,896	5.59×
`" a"`	1,530	1.49×	1,064	1.50×
`"?"`	1,279	5.09×	924	3.59×
`" So"`	901	5.78×	846	4.43×
`" to"`	1,051	1.46×	594	1.26×
`"2"`	1,294	0.42×	343	0.24×
`" But"`	720	7.81×	914	7.32×
`" but"`	733	5.51×	859	5.51×

so $F(v) > 1$ means that v precedes repairs more often than its base rate. Table 9 lists the most frequent repair-preceding tokens, and Table 10 lists the tokens with the highest $F(v)$. By count, repairs are most often preceded by punctuation, paragraph breaks, and function words (".", ",", "the", "is"). By relative frequency, the most over-represented tokens signal a shift in reasoning: contrast ("But", "but", "Or"), verification ("check", "typo", "calculation", "seems"), and conclusion ("So"). This is consistent with prior findings that high-entropy tokens act as forks in reasoning (Wang et al., 2025a). In contrast, digits such as "2" are under-represented ($F = 0.42$ and 0.24), suggesting that routine computation rarely triggers repair.

A.8 Additional Trajectory Analysis

Case study: repeated verification after obtaining the answer. We examine MATH-500 problem 245(`math500_245`) using Qwen3-4B. The task asks for the remainder of a polynomial $f(x)$ upon division by $x^2 - 1$. Writing the remainder as $ax + b$, evaluating at $x = 1$ and $x = -1$ gives $a + b = -10$ and $-a + b = 16$. Thus, $a = -13$ and $b = 3$, which fully determines the correct remainder, $-13x + 3$.

The excerpts in Figure 6 distinguish obtaining this result during thinking from completing the final response. Both methods obtain the correct remainder within the first 1,600 generated tokens. Quest subsequently revisits the same evaluations and polynomial reductions, and is still checking when it reaches the 32,768-token limit. HERA also verifies its result, but proceeds to a correct final response and terminates after 4,841 tokens. These excerpts illustrate the behavior of the two complete methods, rather than the effect of an isolated repair.

Table 10: **Tokens with the highest relative frequency F.** Tokens appearing fewer than 10 times in either the repair or the non-repair set are excluded. The most over-represented tokens mark shifts in reasoning, such as contrast ("But"), verification ("check", "typo"), and conclusion ("So"). "rotein" is a subword of "protein".

Rank	Preceding token	Repair count	Non-repair count	F
		$h = 0.5$		
1	`"But"`	348	173	9.72×
2	`" But"`	720	446	7.81×
3	`" check"`	193	126	7.40×
4	`" \n\n"`	2,216	1,534	6.99×
5	`" count"`	14	10	6.68×
6	`" calculated"`	21	16	6.30×
7	`" Because"`	23	18	6.15×
8	`"\"."`	39	31	6.07×
9	`" So"`	901	754	5.78×
10	`" Or"`	74	64	5.59×
11	`" should"`	106	93	5.51×
12	`" but"`	733	644	5.51×
		$h = 0.9$		
1	`"But"`	571	794	10.31×
2	`" But"`	914	1,791	7.32×
3	`"rotein"`	11	25	6.46×
4	`" see"`	23	52	6.41×
5	`" should"`	115	267	6.19×
6	`"**\n\n"`	16	39	5.99×
7	`" \n\n"`	1,896	4,864	5.59×
8	`" but"`	859	2,233	5.51×
9	`" typo"`	14	42	4.89×
10	`" seems"`	28	87	4.67×
11	`")."`	354	1,116	4.55×
12	`" calculation"`	21	68	4.50×

A.9 Truncation Rate

Table 11 shows the truncation rates for all models and benchmarks in our study. We found that HERA substantially reduces truncation relative to Quest and often improves over periodic repair. This indicates that HERA is not only better at lowering average length of generation, it also alleviates the long-tailed generation induced by sparse attention.

A.10 Additional Intervention Study on GPQA-Diamond

To test whether the high-entropy sensitivity observed on MATH-500 extends beyond mathematics, we repeat the intervention study of Sec. 3.2 on GPQA-Diamond with Qwen3-8B. We compare sparse-history/sparse-current interventions at positions with $H_t \geq 0.5$ against an approximately rate-matched random control, while otherwise decoding with full attention. We use a Quest budget of 1,024 tokens, a 32,768-token generation limit, and one sampling seed. Both policies intervene at roughly 19% of generated positions.

As shown in Table 12, high-entropy interventions substantially lengthen generation relative to both dense decoding and the random control, whereas random interventions have little effect. The corresponding accuracy differences are small. Together with the MATH-500 results, this provides additional evidence that sparse states at high-entropy decisions have disproportionately large effects on reasoning trajectories.

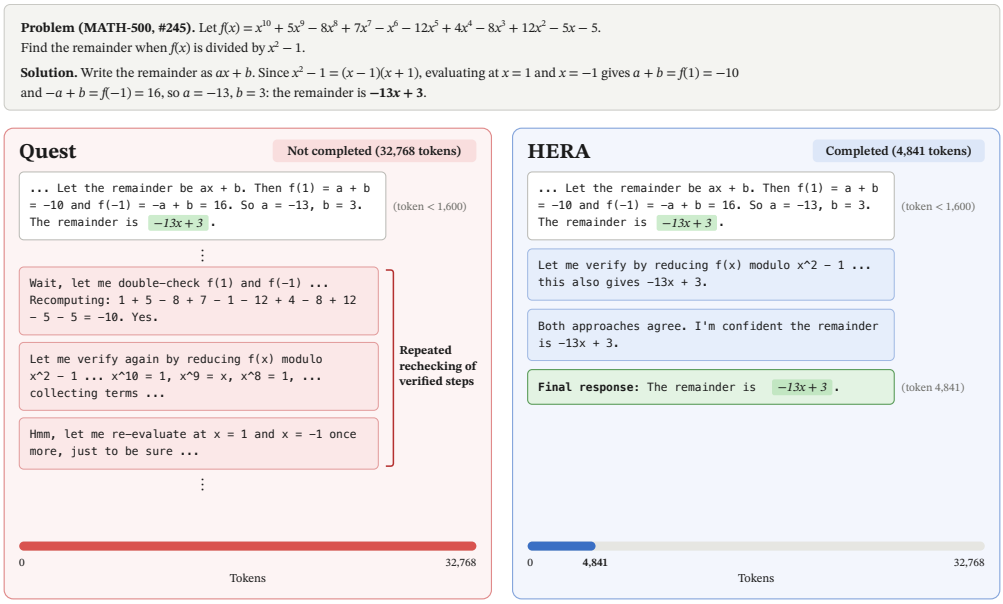


Figure 6: Case study on MATH-500 problem 245 with Qwen3-4B. Both Quest (left) and HERA (right) obtain the correct remainder $-13x + 3$ within the first 1,600 generated tokens. Quest keeps rechecking previously verified steps until it reaches the 32,768-token limit without completing its response; HERA verifies the result once more and completes its response at 4,841 tokens. Excerpts are abridged; bars show generated tokens relative to the limit.

Table 11: Truncation rate (%) of all methods across models and benchmarks, complementing Tables 2–4. Parenthesized values beside HERA denote the entropy threshold h; the first HERA column pairs with ReSA-16 and the second with ReSA-32.

Benchmark	Dense	Quest	ReSA-16	**HERA**	ReSA-32	**HERA**
Qwen3-8B						
MATH-500	8.1	36.8	20.7	15.1 (0.5)	23.9	20.5 (0.9)
AIME 2025	10.0	28.9	16.7	11.1 (0.5)	16.7	17.8 (0.9)
GPQA-Diamond	0.0	14.9	1.5	0.3 (0.5)	1.0	1.3 (1.1)
LiveCodeBench v6	2.6	53.4	14.6	12.6 (0.9)	23.4	16.6 (1.05)
Qwen3-4B						
MATH-500	0.0	25.0	8.2	3.4 (0.5)	11.0	7.0 (0.9)
AIME 2025	10.0	23.3	20.0	10.0 (0.5)	20.0	13.3 (0.9)
GPQA-Diamond	1.0	11.6	0.5	0.5 (0.7)	2.5	2.0 (1.1)
LiveCodeBench v6	2.9	56.0	17.1	8.0 (0.7)	27.4	21.1 (1.1)
DeepSeek-R1-Distill-Llama-8B						
MATH-500	0.6	12.7	1.9	1.1 (0.7)	3.4	2.0 (1.3)
AIME 2025	5.0	11.7	5.0	3.3 (0.9)	8.3	3.3 (1.2)
GPQA-Diamond	5.1	8.1	0.0	0.0 (0.8)	0.8	0.5 (1.3)
LiveCodeBench v6	6.9	27.4	0.3	0.3 (0.9)	0.9	0.9 (1.35)

Table 12: **High-entropy interventions lengthen generations on GPQA-Diamond far more than rate-matched random interventions.** Results on GPQA-Diamond with Qwen3-8B and Quest (token budget 1,024) with 32,768-token generation limit. Sparse attention is applied only at intervened steps, while all other steps use full attention. High-entropy interventions are selected by $H_t \geq 0.5$; random interventions are approximately matched to the same intervention rate. Rate is the fraction of intervened steps.

Policy	Acc. (%)	Tokens	Trunc. (%)	Rate (%)
Dense	61.62	9,825	0.00	–
High entropy	60.10	12,508	1.52	19.74
Matched random	62.12	10,226	0.51	19.25