Thought-Aware KV Cache Compaction for Reasoning via Adaptive Attention Matching

Anonymous ACL submission

Abstract

Reasoning language models generate lengthy chain-of-thought (CoT) sequences whose key-value (KV) cache grows linearly and becomes a memory bottleneck during decoding. Existing compaction methods treat reasoning trajectories as flat token sequences and apply uniform compression, ignoring the hierarchical structure of CoT reasoning where different steps vary drastically in importance. We propose **Thought-Aware Attention Matching (TAM)**, which exploits this structure through three mechanisms: (i) thought segmentation that decomposes the trajectory into reasoning blocks, (ii) adaptive budget allocation that assigns compression budget based on each segment's importance and size, and (iii) pivotal token protection that preserves high-attention reasoning anchors. We prove that the allocation rule is optimal under a convex error model and that cumulative error under sequential compaction remains bounded. Experiments on AIME 2024 and MATH-500 with Qwen3-4B show that TAM improves accuracy over uniform compaction at the same memory footprint, with periodic compaction bounding peak memory to 3.1–3.2 GB (a 65% reduction) while maintaining competitive accuracy.

1 Introduction

Memory has emerged as a critical bottleneck in modern language models deployed for long-horizon tasks. Reasoning-capable models produce lengthy chain-of-thought (CoT) sequences (Wei et al., 2022) before final answers, and the key-value (KV) cache that stores attention states for all previous tokens grows linearly with sequence length (Zhang et al., 2023). This growth substantially increases memory footprint and reduces decoding throughput (Dao et al., 2022), limiting the practical deployment of reasoning models on resource-constrained devices. While existing KV cache compression methods (Zhang et al., 2023; Li et al., 2024) primarily target long input prompts at prefill time, the efficient compaction of *generated* tokens during decoding, i.e. mid-trajectory compaction, remains underexplored despite its importance for reasoning workloads that routinely produce long chains of intermediate tokens.

Recent work has made progress along complementary directions. Attention Matching (AM) (Zweiger et al., 2026) achieves large one-shot compaction by constructing compact keys and values that reproduce attention outputs and preserve attention mass, using closed-form solutions that avoid gradient descent. However, AM targets prefill scenarios and assumes reference queries from self-study on a fixed context. Reasoning Path Compression (RPC) (Song et al., 2025) addresses mid-trajectory compression by periodically evicting tokens based on a selector window of recent queries. Yet RPC relies on simple eviction, which, as we show formally, systematically underestimates attention mass at high compression ratios. Both approaches treat the reasoning trajectory as a flat token sequence and apply uniform compression, ignoring the hierarchical structure of CoT reasoning. While prior methods vary along two axes—the compaction primitive (eviction vs. optimization) and the query source (self-study vs. selector window)—neither exploits a third, orthogonal axis: structure-aware budget allocation.

CoT trajectories (Kojima et al., 2022) naturally decompose into distinct reasoning steps—problem restatements, intermediate calculations, exploratory dead ends, and conclusions—whose importance to future generation varies dramatically. Dead-end explorations become irrelevant as reasoning progresses, while key intermediate results and problem definitions remain critical throughout. This heterogeneity motivates non-uniform budget allocation: by assigning more compression budget to important reasoning steps and less to irrelevant

ones, we can preserve the tokens that matter most while aggressively compressing the rest.

We propose **Thought-Aware Attention Matching (TAM)**, which realizes this idea through three mechanisms: (i) thought segmentation that decomposes the reasoning trajectory into coherent blocks, (ii) adaptive budget allocation that distributes compact keys based on each segment's importance and size, and (iii) pivotal token protection that preserves high-attention reasoning anchors. These mechanisms operate on top of the AM optimization pipeline and a selector window for lightweight query generation, transforming uniform compaction into a structure-aware scheme.

We make three contributions:

1. We introduce TAM, a structure-aware mid-trajectory KV cache compaction method that segments reasoning trajectories into thought blocks, allocates compression budget adaptively based on each segment's importance and size, and protects critical reasoning anchors via pivotal token identification.

2. We provide theoretical foundations: the allocation $t_i \propto \sqrt{w_i \cdot n_i}$ is optimal under convex error models (Proposition C.1), pivotal protection reduces approximation error (Proposition C.2), and cumulative error under sequential compaction remains bounded (Proposition C.4).

3. We evaluate TAM on AIME 2024 and MATH-500 with Qwen3-4B, demonstrating consistent accuracy improvements over uniform compaction at the same memory budget, with periodic compaction achieving up to 65% peak-memory reduction.

2 Related Work

KV cache eviction and merging. KV cache compression has been studied extensively for long-context inference (Xiao et al., 2023; Bai et al., 2024; Liu et al., 2024; Ding et al., 2023). H2O (Zhang et al., 2023) retains tokens with the highest accumulated attention under a sliding window; SnapKV (Li et al., 2024) selects important KV positions by observing attention patterns within an observation window. These methods target long input prompts at prefill time and are not designed for dynamically growing context during decoding. All eviction-based methods apply uniform retention policies, treating every token position equivalently.

Latent-space compaction. Cartridges (Eyuboglu et al., 2025) train compact KV caches offline via prefix-tuning at the cost of GPU-hours per context. Attention Matching (AM) (Zweiger et al., 2026) decomposes the compaction problem into closed-form subproblems (OMP for key selection, NNLS for biases, OLS for value fitting), matching Cartridges quality at far lower cost but assuming one-shot compaction at prefill time. Both methods apply uniform compression across all tokens.

Mid-trajectory compression for reasoning. Reasoning Path Compression (RPC) (Song et al., 2025) addresses mid-trajectory compression by periodically evicting tokens based on a selector window of recent queries. While RPC identifies the selector window as a lightweight proxy for future queries, it relies on eviction rather than latent-space optimization and applies a uniform retention threshold across the entire prefix. TAM operates along an orthogonal axis: it allocates compression budget non-uniformly across reasoning segments based on their importance, a design dimension independent of the compaction primitive or query source.

3 Preliminaries

3.1 Attention and KV Cache

In the Transformer architecture (Vaswani et al., 2017), each layer computes attention over queries $\mathbf{Q} \in \mathbb{R}^{n \times d}$, keys $\mathbf{K} \in \mathbb{R}^{T \times d}$, and values $\mathbf{V} \in \mathbb{R}^{T \times d}$. The scaled dot-product attention is

$$\mathrm{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V}, \quad (1)$$

where the softmax is applied row-wise. For autoregressive generation, the model caches keys and values for all T context tokens, appending new KV pairs as each token is generated. The KV cache thus grows linearly with sequence length, becoming a memory bottleneck for long reasoning trajectories (Dao et al., 2022).

A key insight for compaction is that attention over concatenated KV blocks decomposes into a mixture of each block's locally normalized output, weighted by that block's attention mass. For a key block $\mathbf{K}$, define the *attention mass* as

$$\mathrm{Mass}(\mathbf{q}; \mathbf{K}) = \sum_{j=1}^{T} \exp\left(\frac{\mathbf{q}\mathbf{K}_j^\top}{\sqrt{d}}\right). \quad (2)$$

Matching both the local attention output and the attention mass of a compacted block suffices to preserve its contribution when concatenated with arbitrary future tokens (Zweiger et al., 2026).

3.2 Attention Matching (AM)

Attention Matching (Zweiger et al., 2026) replaces the original cache $(\mathbf{K}, \mathbf{V})$ with $\mathbf{K}, \mathbf{V} \in \mathbb{R}^{T \times d}$ by a compact representation $(\mathbf{C}_k, \boldsymbol{\beta}, \mathbf{C}_v)$ where $\mathbf{C}_k, \mathbf{C}_v \in \mathbb{R}^{t \times d}$ with $t \ll T$, and $\boldsymbol{\beta} \in \mathbb{R}^t$ is a per-token scalar bias. The compact cache should satisfy, for reference queries $\mathbf{q}_i$ of interest,

$$\frac{\exp(\mathbf{q}_i \mathbf{K}^\top / \sqrt{d})\mathbf{V}}{\sum_j \exp(\mathbf{q}_i \mathbf{K}_j^\top / \sqrt{d})} \approx \frac{\exp(\mathbf{q}_i \mathbf{C}_k^\top / \sqrt{d} + \boldsymbol{\beta})}{\sum_j \exp(\mathbf{q}_i (\mathbf{C}_k)_j^\top / \sqrt{d} + \beta_j)} \mathbf{C}_v, \tag{3}$$

$$\sum_{j=1}^{T} \exp(\mathbf{q}_i \mathbf{K}_j^\top / \sqrt{d}) \approx \sum_{j=1}^{t} \exp(\mathbf{q}_i (\mathbf{C}_k)_j^\top / \sqrt{d} + \beta_j). \tag{4}$$

The first condition matches the local attention output; the second matches the attention mass. With the substitution $u_j = \exp(\beta_j) \geq 0$, the mass condition becomes a nonnegative least squares problem over $\mathbf{u}$, solved via NNLS (**?**). Given $\mathbf{C}_k$ and $\boldsymbol{\beta}$, the value matrix $\mathbf{C}_v$ is fitted via ordinary least squares. Key selection is performed via orthogonal matching pursuit (Tropp and Gilbert, 2007) (which greedily selects keys to minimize mass residual) or by retaining keys with highest aggregate attention under the reference queries. In prefill AM, reference queries are obtained from self-study or repeat-prefill on the fixed context.

4 Thought-Aware Attention Matching (TAM)

4.1 Overview

TAM transforms uniform compaction into a structure-aware scheme through three mechanisms operating on top of the AM optimization pipeline: (i) thought segmentation, (ii) adaptive budget allocation, and (iii) pivotal token protection. Section 4.8 provides theoretical justification for the allocation strategy; Figure 1 illustrates the full pipeline.

4.2 Selector Window Query Generation

Let the current sequence length be L, comprising the prompt and all reasoning tokens generated so far. The **selector window** comprises the last R tokens. We extract query vectors $\mathbf{Q}_{\text{ref}} \in \mathbb{R}^{R \times d}$ from these positions by running a single forward pass with hooks to capture query activations. This

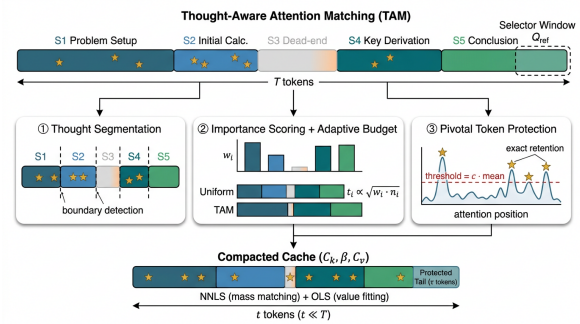


Figure 1: Overview of the TAM pipeline.

yields per-layer, per-head reference queries; TAM processes each KV-head independently using its corresponding $\mathbf{Q}_{\text{ref}}$. These queries encode the current reasoning frontier and serve as a lightweight proxy for future query distributions without requiring self-study or repeat-prefill (Song et al., 2025).

4.3 Thought Segmentation

TAM decomposes the compactable prefix into reasoning segments $\{S_1, S_2, \ldots, S_m\}$, where each segment corresponds to a coherent reasoning step.

Heuristic segmentation. Chain-of-thought outputs exhibit clear structural patterns that delineate reasoning boundaries, most prominently paragraph breaks (double newlines). Reasoning models are trained to organize their output into coherent steps separated by such structural markers, so these boundaries reliably correspond to semantic transitions. We detect double-newline boundaries in the decoded token text using a lightweight rule-based scanner that runs in $O(L)$ time. Consecutive short segments (fewer than $\ell_{\min}$ tokens) are merged to avoid over-fragmentation.

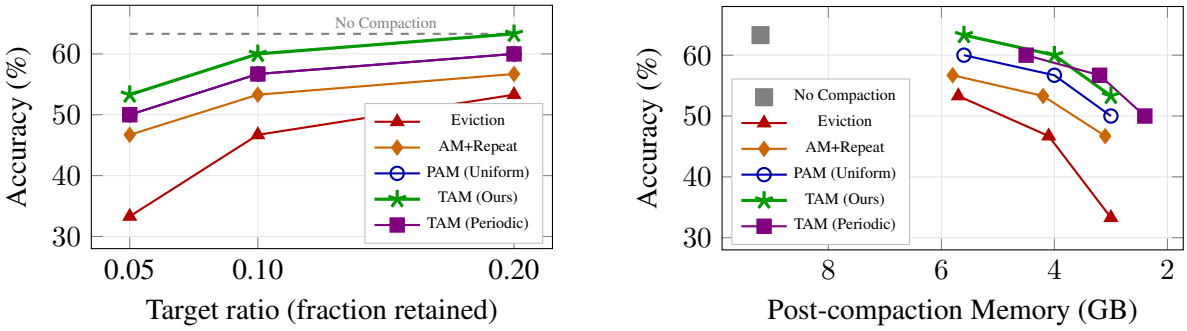
Attention-based segmentation (alternative). We also evaluate an attention-based approach that detects boundaries from shifts in attention intensity. For each position j, we compute the mean attention $\bar{\alpha}_j$ it receives from the selector window (defined formally in Eq. 7). We then measure the local gradient $\Delta_j = |\bar{\alpha}_j - \bar{\alpha}_{j-1}|$ and place boundaries at positions where Δ_j exceeds $2\times$ the median gradient. This detects transitions between high-attention and low-attention regions, which empirically correspond to reasoning step boundaries. We compare both approaches in Section 5.4.

Table 1: Main results at target compaction ratio 0.1 (retain 10% of KV entries). "Mem" is post-compaction steady-state GPU memory (see Section 5.1); for one-shot methods ($P{=}\infty$), peak memory during generation equals No Compaction. Only TAM (Periodic) maintains bounded memory throughout. Best accuracy in **bold**; best memory in underline.

Method	AIME 2024 (30 prob.)		MATH-500	
	Acc (%)	Mem (GB)	Acc (%)	Mem (GB)
No Compaction	63.3	9.2	71.2	8.8
Eviction (Selector Window)	46.7	4.1	52.4	3.9
AM + Repeat	53.3	4.2	61.0	4.0
PAM (Uniform)	56.7	4.0	64.6	3.8
TAM (Ours)	**60.0**	4.0	**67.8**	3.8
TAM (Periodic, $P{=}1024$)	56.7	<u>3.2</u>	65.4	<u>3.1</u>

Table 2: TAM compaction time breakdown per step (Qwen3-4B, ∼4k tokens). TAM-specific stages are marked with †.

Stage	Time (s)
Query extract (selector window)	0.8
Thought segmentation + importance†	0.1
Pivotal token identification†	<0.1
Per-segment key selection	2.5
β fitting (NNLS, global)	1.2
Value fitting (OLS, global)	0.9
Total	**5.6**
of which TAM-specific	*∼0.15*



(a) Accuracy vs. compaction ratio (AIME 2024).

(b) Accuracy vs. post-compaction memory (AIME 2024).

Figure 2: Accuracy vs. compaction trade-offs on AIME 2024 (Qwen3-4B). Left: accuracy vs. compaction ratio. Right: accuracy vs. post-compaction memory. TAM consistently outperforms all baselines; TAM (Periodic) achieves the best memory vs. accuracy trade-off.

4.4 Segment Importance Scoring and Adaptive Budget Allocation

Given segments $\{S_1, \ldots, S_m\}$ with sizes $n_1, \ldots, n_m$ and selector window queries $\mathbf{Q}_{\text{ref}}$, we compute the *importance* of each segment as the average attention mass it receives from the current reasoning frontier:

$$w_i = \frac{1}{|\mathbf{Q}_{\text{ref}}|} \sum_{\mathbf{q} \in \mathbf{Q}_{\text{ref}}} \sum_{j \in S_i} \alpha_j(\mathbf{q}), \tag{5}$$

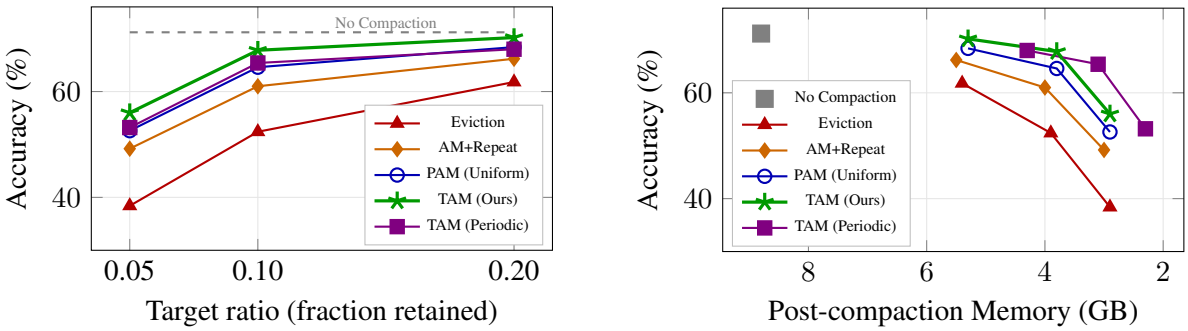
where $\alpha_j(\mathbf{q})$ is the softmax attention weight of token j under query $\mathbf{q}$ (normalized over the full prefix including the protected tail). Since the sum only covers compactable positions, $\sum_i w_i \leq 1$; the residual is the tail's attention mass, which is typically small.

Optimal budget allocation. Let $\mathcal{P}$ denote the set of pivotal tokens (Section 4.5) and $t' = t - |\mathcal{P}|$ the remaining budget after reserving their slots. We allocate t_i compact keys to segment S_i according to:

$$t_i = \max\left(1, \left\lfloor t' \cdot \frac{\sqrt{w_i \cdot n_i}}{\sum_{j=1}^{m} \sqrt{w_j \cdot n_j}} \right\rfloor\right). \tag{6}$$

Any residual budget from rounding is assigned to the segment with the largest fractional remainder. This allocation has a principled justification: in Proposition C.1, we prove that when the per-segment approximation error is a convex function of the local compression ratio n_i/t_i, the importance-weighted allocation $t_i \propto \sqrt{w_i \cdot n_i}$ minimizes the total weighted approximation error under a linear error model. The allocation naturally concentrates budget on segments that are both large and important, while assigning minimal budget to irrelevant segments.

Standard AM and its uniform variant with selector window queries (PAM, for Prefix Attention Matching) use $t_i \propto n_i$, which wastes budget on low-importance segments. Pure attention-based allocation $t_i \propto w_i$ ignores segment size, potentially under-representing large segments. TAM's



(a) Accuracy vs. compaction ratio (MATH-500).

(b) Accuracy vs. post-compaction memory (MATH-500).

Figure 3: Accuracy vs. compaction trade-offs on MATH-500 (Qwen3-4B, 500 problems). Left: accuracy vs. compaction ratio. Right: accuracy vs. post-compaction memory. With larger sample size, TAM's advantage over PAM is more clearly separated. TAM achieves near-full-accuracy recovery at ratio 0.2 (70.2% vs. 71.2%).

Table 3: Component ablation at target ratio 0.1.

Configuration	AIME 2024		MATH-500	
	Acc (%)	Δ	Acc (%)	Δ
PAM (Uniform)	56.7	0	64.6	0
+ Adaptive allocation	56.7	+0.0	66.4	+1.8
+ Pivotal protection	56.7	+0.0	66.2	+1.6
+ Both (**TAM**)	**60.0**	**+3.3**	**67.8**	**+3.2**

Table 4: Ablation Study (TAM, target ratio 0.1, AIME 2024).

Segmentation Method	**Acc (%)**	**Seg. Time (ms)**
Heuristic (double-newline)	**60.0**	2
Attention-based (gradient jump)	60.0	45
Fixed-length (ℓ=256)	56.7	1

$\sqrt{w_i \cdot n_i}$ allocation balances both factors optimally.

4.5 Pivotal Token Protection

Beyond the fixed tail protection of the last τ tokens, TAM identifies and protects *pivotal tokens*: positions that receive consistently high attention from the selector window, indicating their role as reasoning anchors (e.g., problem constants, key intermediate results, critical definitions).

For each token j in the compactable prefix, we compute its average attention score:

$$\bar{\alpha}_j = \frac{1}{|\mathbf{Q}_{\text{ref}}|} \sum_{\mathbf{q} \in \mathbf{Q}_{\text{ref}}} \alpha_j(\mathbf{q}). \tag{7}$$

A token is pivotal if $\bar{\alpha}_j > \delta$, where we set $\delta = c \cdot \bar{\alpha}_{\text{mean}}$ with $c = 3$ by default, selecting tokens whose attention significantly exceeds the mean. The pivotal set $\mathcal{P}$ is constructed by collecting all such tokens; their keys and values are directly retained in the compact cache without modification. The remaining budget $t' = t - |\mathcal{P}|$ is distributed across segments via Eq. 6, and key selection within each segment operates only on non-pivotal tokens.

In Proposition C.2, we show that pivotal protection reduces the worst-case approximation error by a factor proportional to the total attention mass of pivotal tokens.

4.6 Periodic Compaction Trigger

Compaction is triggered when the cache reaches the maximum sequence length or P new tokens have been generated since the last compaction. At each trigger, TAM re-runs the full pipeline on the current cache: segment boundaries are recomputed from scratch on the updated token sequence, importance scores are recalculated using a fresh selector window, and pivotal tokens are re-identified. Previously protected pivotal tokens receive no special treatment in subsequent rounds—they compete for retention based on their current attention scores. The target size t remains fixed across rounds, so the cache oscillates between t and $t + P$ tokens. The parameter P trades off peak memory against compaction overhead: smaller P compacts more frequently, keeping memory bounded at the cost of higher compute; larger P amortizes overhead but allows higher peak memory. Proposition C.4 quantifies how P controls cumulative error.

4.7 Algorithm

Algorithm 1 summarizes one TAM compaction step. After protecting the last τ tokens and extracting $\mathbf{Q}_{\text{ref}}$, TAM segments the compactable prefix, identifies pivotal tokens, allocates per-segment budgets, and runs key selection within each segment. While key selection is per-segment, the bias fit-

Table 5: Ablation on pivotal threshold multiplier c (TAM, target ratio 0.1, AIME 2024).

Threshold c	$\lvert\mathcal{P}\rvert$ **(avg)**	**Acc (%)**
1	312	56.7
3 (default)	87	**60.0**
5	34	60.0
10	8	56.7

Table 6: Ablation on compaction interval P (TAM, target ratio 0.1, AIME 2024).

Interval P	**Acc (%)**	**Peak Mem (GB)**
∞ (max only)	**60.0**	4.0
1024 (default)	56.7	3.2
512	56.7	3.0
256	53.3	**2.8**

ting (NNLS) and value fitting (OLS) are performed *globally* across all selected keys, ensuring that attention mass and outputs are matched at the full-prefix level. To guarantee exact retention of pivotal tokens (as required by Proposition C.2), we fix $\beta_j = 0$ for all pivotal keys $j \in \mathcal{P}$, so their original logits are preserved unchanged; NNLS only optimizes biases for non-pivotal selected keys.

4.8 Theoretical Analysis

We provide principled justification for TAM's design choices. Proofs are deferred to Appendix D.

Proposition 4.1 (Mass Deficit of Eviction)**.** *Consider eviction-based compaction that retains a subset $S \subset [T]$ with $|S| = t < T$ keys and values without scalar biases. For any query $\mathbf{q}$, define the* evicted mass fraction

$$\mu(\mathbf{q}) = \frac{\sum_{j \notin S} \exp(\mathbf{q}\mathbf{K}_j^\top/\sqrt{d})}{\sum_{j=1}^{T} \exp(\mathbf{q}\mathbf{K}_j^\top/\sqrt{d})} = \sum_{j \notin S} \alpha_j(\mathbf{q}), \tag{8}$$

where $\alpha_j(\mathbf{q})$ is the original attention weight of token j. Then:

1. *$\mathit{Mass}(\mathbf{q}; \mathbf{C}_k) = (1 - \mu(\mathbf{q})) \cdot \mathit{Mass}(\mathbf{q}; \mathbf{K}) < \mathit{Mass}(\mathbf{q}; \mathbf{K})$ for all $\mathbf{q}$ whenever $S \neq [T]$.*
2. *When the compacted block is concatenated with future KV pairs $(\mathbf{K}_f, \mathbf{V}_f)$, the compacted prefix's contribution to the softmax denominator is deflated by factor $(1 - \mu(\mathbf{q}))$, shifting attention toward $(\mathbf{K}_f, \mathbf{V}_f)$.*
3. *Considering the prefix block in isolation, the attention output error satisfies $\|\hat{\mathbf{y}} - \mathbf{y}\| \leq 2\mu(\mathbf{q}) \cdot \|\mathbf{V}\|_\infty$, where $\|\mathbf{V}\|_\infty = \max_j \|\mathbf{V}_j\|$.*

Proposition 4.1 formalizes the fundamental limitation of eviction: the mass deficit $\mu(\mathbf{q})$ equals the total attention weight of evicted tokens, which can be substantial even when each individual evicted token has low attention (the "long tail" of small weights sums to a non-negligible fraction). TAM addresses this by fitting scalar biases $\boldsymbol{\beta}$ via NNLS to preserve mass for the reference queries.

5 Experiments

We evaluate TAM on mathematical reasoning benchmarks, comparing against uniform compaction baselines and eviction methods. We measure downstream accuracy, peak memory usage, and compaction time, and provide ablation studies on TAM's key components.

5.1 Experimental Setup

Benchmarks and model. We evaluate on two benchmarks: (1) **AIME 2024** (Zhang and Math-AI, 2024) (AIME I and AIME II, 30 problems total), which requires multi-step mathematical reasoning and produces lengthy CoT trajectories (median generation length ≈4k tokens); and (2) **MATH-500**, a 500-problem subset of the MATH test set (Hendrycks et al., 2021; Cobbe et al., 2021) spanning algebra, geometry, number theory, and combinatorics, providing larger-scale evaluation with greater statistical power. Each problem has a deterministic answer; we use Qwen3-4B (Yang et al., 2025) as the base model with greedy decoding (temperature 0) and a maximum generation length of 8192 tokens. Accuracy is measured by exact match (pass@1) (Chen et al., 2021).

Compared methods. We compare six configurations: (1) **No Compaction**: full KV cache (accuracy upper bound); (2) **Eviction (Selector Window)**: retains high-attention keys and evicts the rest (Song et al., 2025); (3) **AM + Repeat**: one-shot AM at max length with repeat-prefill queries (Zweiger et al., 2026); (4) **PAM (Uniform)**: AM with selector window queries and uniform budget (no segmentation or pivotal protection), compacting once at max length ($P = \infty$); (5) **TAM (Ours)**: AM with selector window queries, segmentation, adaptive budget, and pivotal protection, compacting once at max length ($P = \infty$); (6) **TAM (Periodic)**: TAM with periodic triggers every $P = 1024$ tokens for bounded peak memory.

We note that H2O (Zhang et al., 2023) and SnapKV (Li et al., 2024) are prefill-oriented evic-

tion methods designed for long input contexts. Our Eviction (Selector Window) baseline adapts their core mechanism—attention-based token selection—to the mid-trajectory setting, providing a representative comparison for eviction-based approaches in the decoding scenario.

Evaluation protocol. We sweep the target compaction ratio over $\{0.05, 0.1, 0.2\}$ (retain 5%, 10%, or 20% of KV entries). We protect the last τ=20 tokens, use a selector window of R=64 tokens, and set the pivotal threshold multiplier c=3. Thought segmentation uses the heuristic (double-newline) method by default with $\ell_{\min}$=32. Key selection uses the highest-attention heuristic. All experiments use a single NVIDIA A100 GPU (80GB). The model is loaded in 4-bit quantization (GPTQ-Int4), so model weights occupy ≈2.5 GB; the reported "Mem (GB)" is the *post-compaction steady-state memory* measured via `torch.cuda.max_memory_allocated` after the first compaction step completes, averaged over all problems. For one-shot methods (P=∞), this reflects memory after the single compaction at max sequence length; the transient peak before compaction is comparable to No Compaction. For TAM (Periodic), memory stays bounded throughout generation.

5.2 Main Results

Table 1 reports results at target ratio 0.1 on both benchmarks. Figure 2a and Figure 3a sweep the compaction ratio for AIME and MATH-500 respectively; Figure 2b and Figure 3b plot the accuracy vs. memory trade-off. We highlight five findings.

First, the Eviction baseline suffers a substantial accuracy drop on both benchmarks (46.7% on AIME, 52.4% on MATH-500 vs. 63.3% and 71.2%), confirming Proposition 4.1: mass deficit degrades generation quality. Second, PAM (Uniform) improves over eviction by applying the AM pipeline but still falls short of No Compaction. Third, **TAM consistently outperforms PAM (Uniform)** on both benchmarks (60.0% vs. 56.7% on AIME; 67.8% vs. 64.6% on MATH-500 at ratio 0.1), demonstrating that thought-aware allocation and pivotal protection provide genuine gains over uniform compression. Fourth, TAM (Periodic) achieves the best steady-state memory (3.2 GB, 65% reduction vs. No Compaction) while maintaining competitive accuracy. On MATH-500, TAM (Periodic) outperforms PAM by 0.6–0.8 points at ratios 0.05–0.1, confirming that TAM's per-step advantage partially survives cumulative error from repeated compaction. At ratio 0.2, cumulative error slightly offsets TAM's advantage on MATH-500 (68.0% vs. 68.4% for PAM), consistent with diminishing returns of adaptive allocation when ample budget reduces the impact of importance heterogeneity. Fifth, at ratio 0.2, TAM recovers the full No Compaction accuracy on AIME (63.3%); on MATH-500, a modest 1.0-point gap remains (70.2% vs. 71.2%).

Statistical significance. AIME 2024 has only 30 problems (3.3% per-problem granularity), so individual accuracy differences (e.g., TAM vs. PAM: 60.0% vs. 56.7%, a single-problem difference) are not statistically significant by themselves and should be interpreted with caution. In particular, TAM (Periodic) and PAM are statistically indistinguishable on AIME at all ratios; MATH-500 provides the meaningful comparison for periodic compaction. On MATH-500, TAM vs. PAM (67.8% vs. 64.6%, $\Delta = 16$ problems) yields a one-sided McNemar test $p \approx 0.04$, providing moderate evidence of improvement. The consistent direction of improvement across both benchmarks and all three compression ratios (Table 7 and Table 8) strengthens the case despite AIME's limited statistical power. We recommend that future work use larger evaluation sets and report confidence intervals; we note this limitation in Section 6.

5.3 Compaction Time Breakdown

Table 2 profiles TAM's wall-clock cost per compaction step on a typical ∼4k-token trajectory. Thought segmentation and importance scoring add minimal overhead (0.1 s total) relative to the AM core (key selection + fitting). PAM (Uniform) takes approximately 5.4 s per step (the same AM core without the 0.15 s structure-aware overhead), confirming that TAM's additional cost is negligible. For context, generating a ∼4k-token trajectory takes 60–80 s on a single A100, so the compaction cost represents less than 10% of total inference time.

5.4 Ablation Studies

We conduct ablation studies on TAM's key components: adaptive budget allocation, pivotal token protection, segmentation method, the compaction interval P, and the selector window size R. All

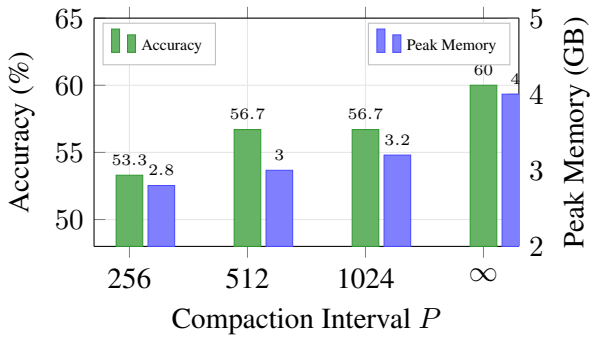


Figure 4: Ablation on compaction interval P (AIME 2024). Smaller P reduces peak memory with modest accuracy trade-off. $P{=}\infty$ compacts only at max sequence length.

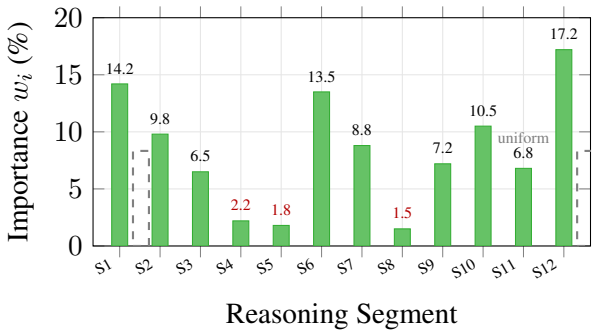


Figure 5: Per-segment importance w_i for a representative AIME problem (12 segments). Importance varies by $>11\times$ (S12: 17.2% vs. S8: 1.5%). Dead-end segments (S4, S5, S8, red labels) receive minimal budget under TAM's adaptive allocation.

ablations use AIME 2024 at target ratio 0.1 unless otherwise noted.

Adaptive allocation and pivotal protection. Table 3 isolates the contribution of each TAM component. On MATH-500 (500 problems, finer granularity), adding adaptive allocation alone improves accuracy by 1.8 points over PAM (Uniform); adding pivotal protection alone yields a 1.6-point gain. The combined gain (+3.2) is slightly less than the sum of individual gains (1.8 + 1.6 = 3.4), indicating a modest overlap: both mechanisms partially address the same error sources, as pivotal tokens (which receive exact retention) also tend to reside in high-importance segments that benefit from adaptive allocation. On AIME (30 problems, 3.3% granularity), individual components do not cross the discrete accuracy threshold, but combining both yields a clear 3.3-point improvement (56.7%→60.0%). On both benchmarks, the full TAM combination outperforms either component in isolation, confirming that the two mechanisms are complementary despite their partial overlap.

Segmentation method. Table 4 compares heuristic segmentation (double-newline boundary detection) with attention-based segmentation (attention gradient jump detection). Both achieve comparable accuracy, with heuristic segmentation being slightly more robust and substantially faster. We use heuristic segmentation as the default.

Pivotal threshold c. Table 5 varies the threshold multiplier c for pivotal token identification. Too low a threshold ($c{=}1$) protects too many tokens, leaving insufficient budget for adaptive allocation. Too high ($c{=}10$) protects too few tokens, reducing the benefit. We find $c{=}3$ to strike a good balance.

Compaction interval P and selector window R. Table 6 ablates the compaction interval. Smaller P compacts more frequently, reducing peak memory; per Proposition C.4, larger P increases λ_k and limits error accumulation. We find $P{=}1024$ a reasonable balance. Table 9 (Appendix) shows that $R{=}64$ is sufficient for the selector window.

6 Conclusion

We introduced Thought-Aware Attention Matching (TAM), which transforms KV cache compaction from a flat, uniform operation into a structure-aware process that exploits the hierarchical nature of chain-of-thought reasoning. Through thought segmentation, adaptive budget allocation, and pivotal token protection, TAM concentrates compression budget where it matters most while aggressively compressing irrelevant steps. Our theoretical analysis establishes the optimality of the allocation rule and bounds cumulative error under sequential compaction. Experiments on AIME 2024 and MATH-500 demonstrate consistent improvements over uniform compaction and eviction baselines, with periodic compaction achieving steady-state memory reduction of up to 65%.

Limitations

TAM has two main limitations. First, it relies on heuristic thought segmentation (e.g., double newlines) and a local selector window to estimate segment importance, which can fail for reasoning traces without clear structural boundaries or when long-range dependencies and backtracking occur. Second, empirical evaluation is limited to mathematical reasoning benchmarks (AIME, MATH-500) and a single model (Qwen3-4B), leaving generalization to other domains and larger models untested.

References

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 3119–3137.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

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

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359.

Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, Nanning Zheng, and Furu Wei. 2023. Longnet: Scaling transformers to 1,000,000,000 tokens. *arXiv preprint arXiv:2307.02486*.

Sabri Eyuboglu, Ryan Ehrlich, Simran Arora, Neel Guha, Dylan Zinsley, Emily Liu, Will Tennien, Atri Rudra, James Zou, Azalia Mirhoseini, et al. 2025. Cartridges: Lightweight and general-purpose long context representations via self-study. *arXiv preprint arXiv:2506.06266*.

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

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

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

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173.

Jiwon Song, Dongwon Jo, Yulhwa Kim, and Jae-Joon Kim. 2025. Reasoning path compression: Compressing generation trajectories for efficient llm reasoning. *arXiv preprint arXiv:2505.13866*.

Joel A Tropp and Anna C Gilbert. 2007. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on information theory*, 53(12):4655–4666.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Yifan Zhang and Team Math-AI. 2024. American invitational mathematics examination (aime) 2025. *Wei Zhao, Zhe Li, Yige Li, Ye Zhang, and Junfeng Sun*.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710.

Adam Zweiger, Xinghong Fu, Han Guo, and Yoon Kim. 2026. Fast kv compaction via attention matching. *arXiv preprint arXiv:2602.16284*.

Appendix

A Algorithm Pipeline

Algorithm 1: Thought-Aware Attention Matching (TAM)

Input: KV cache $(\mathbf{K}, \mathbf{V})$ of length L, target size t, selector window size R, protected tail τ, pivotal threshold multiplier c

Output: Compacted cache

Protect last τ tokens; let $\mathbf{K}_p, \mathbf{V}_p$ be their KV states
Extract $\mathbf{Q}_{\text{ref}}$ from last R tokens via prefill with forward hooks
Segment compactable prefix $[0, L-\tau]$ into $\{S_1, \ldots, S_m\}$ via thought boundaries
for *each KV-head* **do**
  Compute per-token attention scores $\bar{\alpha}_j$ under $\mathbf{Q}_{\text{ref}}$ (Eq. 7)
  Identify pivotal set $\mathcal{P} = \{j : \bar{\alpha}_j > c \cdot \bar{\alpha}_{\text{mean}}\}$
  Retain pivotal keys/values: $\mathbf{C}_k^{\mathcal{P}} \leftarrow \mathbf{K}[\mathcal{P}, :]$, $\mathbf{C}_v^{\mathcal{P}} \leftarrow \mathbf{V}[\mathcal{P}, :]$
  Compute segment importances w_i (Eq. 5) and allocate budgets t_i (Eq. 6)
  for *each segment* S_i **do**
    Select t_i keys from $S_i \setminus \mathcal{P}$ via highest-attention under $\mathbf{Q}_{\text{ref}}$
  end
  Let S_{all} = pivotal keys $\cup$ per-segment selected keys
  Set $\mathbf{C}_k \leftarrow \mathbf{K}[S_{\text{all}}, :]$
  Fit $\boldsymbol{\beta}$ via NNLS to match attention mass globally; set $\beta_j = 0$ for $j \in \mathcal{P}$
  Fit $\mathbf{C}_v$ via OLS to match attention outputs globally
  Concatenate $(\mathbf{C}_k, \boldsymbol{\beta}, \mathbf{C}_v)$ with $(\mathbf{K}_p, \mathbf{V}_p)$
end
return compacted cache

B Analysis: Segment Importance and Budget Distribution

To verify that CoT trajectories exhibit the structural heterogeneity motivating TAM, we analyze the segment importance distribution on AIME 2024 problems.

Importance heterogeneity. Figure 5 visualizes the per-segment importance w_i (Eq. 5) for a representative AIME problem with 12 reasoning segments. Importance varies by over $11\times$ across segments: early segments (problem restatement, initial setup) and the most recent segment receive high importance, while middle segments corresponding to dead-end explorations receive very low importance. This confirms that uniform allocation is wasteful: TAM assigns $\leq 3\%$ of budget to the lowest-importance segments while allocating $> 15\%$ to the highest.

Pivotal token semantics. We qualitatively inspect the pivotal tokens identified by TAM (c=3) across AIME problems. Pivotal tokens predominantly correspond to: (1) numerical constants from the problem statement (e.g., coefficients, constraints), (2) key intermediate expressions (e.g., derived equations, variable bindings), and (3) structural tokens that anchor the reasoning flow (e.g., "Therefore", "="). This aligns with the intuition that pivotal tokens serve as reasoning anchors that future queries consistently rely on.

Error analysis. We examine the problems where TAM (ratio 0.1) fails but No Compaction succeeds. On MATH-500, the accuracy gap (67.8% vs. 71.2%) concentrates on problems with two characteristics: (i) very long reasoning chains (>6k tokens) that produce many segments with complex non-local dependencies between distant reasoning steps, and (ii) problems where the model's reasoning path undergoes a late-stage correction that revisits earlier segments. In case (i), the selector window's local coverage (Proposition C.3) becomes less effective for distant segments, increasing $\|\mathbf{x}_{\mathbf{q}}^{\perp}\|$; in case (ii), compaction

discards tokens from segments previously deemed unimportant that later become relevant upon backtracking. These failure modes suggest that dynamic importance re-estimation (updating segment scores as reasoning progresses) is a promising direction for future work.

C More Propositions

Proposition C.1 (Optimality of Importance-Weighted Allocation)**.** *Consider m segments with sizes $n_1, \ldots, n_m$ and importance weights $w_1, \ldots, w_m$ ($\sum_i w_i = 1$). Suppose the per-segment approximation error when allocating t_i compact keys to segment S_i is $\varepsilon_i = \phi(n_i/t_i)$, where $\phi : [1, \infty) \to [0, \infty)$ is convex and increasing with $\phi(1) = 0$. The importance-weighted total error is*

$$\mathcal{E}(\mathbf{t}) = \sum_{i=1}^{m} w_i \cdot \phi(n_i/t_i), \quad \textit{s.t.} \quad \sum_{i=1}^{m} t_i = t,\ t_i \geq 1. \tag{9}$$

Then:

1. *The optimal allocation satisfies $t_i^\star \propto \sqrt{w_i \cdot n_i \cdot \phi'(n_i/t_i^\star)}$ at interior KKT points.*
2. *Under the linear model $\phi(r) = r - 1$ (error proportional to compression ratio), the optimal allocation simplifies to $t_i^\star \propto \sqrt{w_i \cdot n_i}$.*
3. *Uniform allocation ($t_i \propto n_i$) is strictly suboptimal whenever the importance density w_i/n_i is not constant across segments, i.e., when some segments are more important per token than others.*

Proposition C.1 provides the theoretical foundation for TAM's budget allocation (Eq. 6). The linear-model result $t_i^\star \propto \sqrt{w_i \cdot n_i}$ is the geometric mean of the importance-only allocation ($t_i \propto w_i$) and the size-only allocation ($t_i \propto n_i$), optimally trading off both factors. In CoT reasoning, the importance density w_i/n_i varies substantially across segments: dead-end explorations have low density while key intermediate results have high density, making adaptive allocation strictly beneficial.

Remark C.1 (Linear Error Model)**.** The closed-form $t_i^\star \propto \sqrt{w_i n_i}$ relies on the linear model $\phi(r) = r - 1$. AM's actual fitting error may exhibit more complex dependence on compression ratio, but the general KKT result (Part 1) $t_i^\star \propto \sqrt{w_i n_i \phi'(n_i/t_i^\star)}$ holds for any convex ϕ. In practice, the $\sqrt{w_i n_i}$ allocation consistently outperforms both uniform and importance-only allocation (Table 3), suggesting robustness to the specific form of ϕ in the operating range $r \in [5, 20]$ typical of our experiments.

Proposition C.2 (Error Reduction from Pivotal Token Protection)**.** *Let $\mathcal{P}$ denote the set of pivotal tokens and let $\gamma_{\mathbf{q}} = \sum_{j \in \mathcal{P}} \alpha_j(\mathbf{q})$ be the pivotal attention mass for a specific query $\mathbf{q}$. Suppose the remaining $T - |\mathcal{P}|$ tokens are compacted to $t - |\mathcal{P}|$ keys via AM with per-query approximation error ε_{rest} on the non-pivotal portion. Suppose further that the total attention mass is matched for query $\mathbf{q}$, i.e., $\mathit{Mass}(\mathbf{q}; \mathbf{C}_k, \boldsymbol{\beta}) = \mathit{Mass}(\mathbf{q}; \mathbf{K})$. Then the overall attention output error satisfies*

$$\|\hat{\mathbf{y}} - \mathbf{y}^*\| \leq (1 - \gamma_{\mathbf{q}}) \cdot \varepsilon_{rest}. \tag{10}$$

Define the worst-case pivotal mass $\gamma = \min_{\mathbf{q} \in \mathbf{Q}_{ref}} \gamma_{\mathbf{q}}$, which lower-bounds the per-query pivotal mass over all selector-window queries. Then the uniform bound $\|\hat{\mathbf{y}} - \mathbf{y}^*\| \leq (1 - \gamma) \cdot \varepsilon_{rest}$ holds for all $\mathbf{q} \in \mathbf{Q}_{ref}$. More generally, when mass is not matched exactly, let $\Delta\gamma_{\mathbf{q}} = \hat{\gamma}_{\mathbf{q}} - \gamma_{\mathbf{q}}$ denote the pivotal mass mismatch; then

$$\|\hat{\mathbf{y}} - \mathbf{y}^*\| \leq (1 - \gamma_{\mathbf{q}}) \cdot \varepsilon_{rest} + 2\,|\Delta\gamma_{\mathbf{q}}| \cdot \|\mathbf{V}\|_\infty. \tag{11}$$

Since pivotal tokens are retained exactly, they contribute zero approximation error when mass is preserved. The remaining error arises only from the non-pivotal portion, which carries attention mass $(1 - \gamma)$.

Remark C.2 (Mass-matching Condition)**.** The mass condition $\mathrm{Mass}(\mathbf{q}; \mathbf{C}_k, \boldsymbol{\beta}) = \mathrm{Mass}(\mathbf{q}; \mathbf{K})$ is exactly what AM's NNLS fitting enforces for the reference queries $\mathbf{Q}_{\text{ref}}$. For these queries, $\Delta\gamma_{\mathbf{q}} \approx 0$ and the clean bound $(1 - \gamma)\varepsilon_{\text{rest}}$ applies. For future queries that deviate from $\mathbf{Q}_{\text{ref}}$, the residual mass mismatch $|\Delta\gamma_{\mathbf{q}}|$ is controlled by the selector window approximation (Proposition C.3); under the locality of CoT reasoning, this term remains small.

Remark C.3 (Budget Trade-off)**.** Protecting $|\mathcal{P}|$ tokens reduces the budget available for non-pivotal tokens from t to $t-|\mathcal{P}|$, increasing their compression ratio from T/t to $(T-|\mathcal{P}|)/(t-|\mathcal{P}|)$ and potentially raising $\varepsilon_{\text{rest}}$. The net benefit of pivotal protection therefore depends on the relative magnitudes: it is favorable when γ is large enough that $(1-\gamma)\varepsilon_{\text{rest}} < \varepsilon_{\text{all}}$, where ε_{all} is the error of uniform compaction to size t. Under the linear error model $\phi(r) = r - 1$, a sufficient condition is $\gamma > |\mathcal{P}|/t$, i.e., pivotal tokens' attention mass fraction exceeds their budget fraction, a condition readily satisfied in practice since pivotal tokens are selected precisely for their disproportionately high attention (Table 5 shows $|\mathcal{P}| \approx 87$ out of $t \approx 400$, capturing $\gamma \approx 0.35$ of attention mass, well above the $87/400 = 0.22$ threshold).

Proposition C.3 (Selector Window Approximation Error)**.** *Let $\mathbf{Q}_{ref} \in \mathbb{R}^{n\times d}$ be the selector window queries and $\mathbf{C}_k, \boldsymbol{\beta}, \mathbf{C}_v$ be fitted via TAM. Let $\mathbf{X} \in \mathbb{R}^{n\times t}$ have rows $\mathbf{x}_{ref}^{(i)} = softmax(\mathbf{q}_i\mathbf{C}_k^\top/\sqrt{d} + \boldsymbol{\beta})$ and $\mathbf{C}_v^\star = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{Y}$ be the OLS-fitted values. For a future query $\mathbf{q}$ with attention pattern $\mathbf{x}_\mathbf{q} = softmax(\mathbf{q}\mathbf{C}_k^\top/\sqrt{d} + \boldsymbol{\beta})$, decompose $\mathbf{x}_\mathbf{q} = \mathbf{x}_\mathbf{q}^{\parallel} + \mathbf{x}_\mathbf{q}^{\perp}$ where $\mathbf{x}_\mathbf{q}^{\parallel}$ lies in the row space of $\mathbf{X}$ and $\mathbf{x}_\mathbf{q}^{\perp}$ is orthogonal. Then:*

$$\|\mathbf{x}_\mathbf{q}\mathbf{C}_v^\star - \mathbf{y}^*\| \le \|\mathbf{x}_\mathbf{q}^{\perp}\| \cdot \|\mathbf{C}_v^\star\|_{op} + \|\mathbf{x}_\mathbf{q}^{\parallel}\mathbf{C}_v^\star - \mathbf{y}_{\parallel}^*\|, \tag{12}$$

where $\mathbf{y}^*$ is the ideal output and $\mathbf{y}_{\parallel}^* = \boldsymbol{\lambda}^\top\mathbf{Y}$ with $\boldsymbol{\lambda}$ satisfying $\mathbf{x}_\mathbf{q}^{\parallel} = \boldsymbol{\lambda}^\top\mathbf{X}$ (i.e., the linear combination of reference outputs $\mathbf{Y}$ with the same coefficients that express $\mathbf{x}_\mathbf{q}^{\parallel}$ in the row space of $\mathbf{X}$). When $\mathbf{x}_\mathbf{q}^{\perp} = \mathbf{0}$, the term $\|\mathbf{x}_\mathbf{q}^{\perp}\| \cdot \|\mathbf{C}_v^\star\|_{op}$ vanishes.

The error is controlled by how far the future query's attention pattern deviates from the subspace spanned by the reference patterns. The first term ($\|\mathbf{x}_\mathbf{q}^{\perp}\| \cdot \|\mathbf{C}_v^\star\|_{\text{op}}$) captures coverage error from the selector window's finite span; characterizing it precisely requires distributional assumptions on future queries. Under the locality of CoT reasoning, successive tokens attend to similar context, keeping $\|\mathbf{x}_\mathbf{q}^{\perp}\|$ small. TAM's adaptive allocation further helps: segments where the selector window provides poor coverage (high $\|\mathbf{x}_\mathbf{q}^{\perp}\|$, typically distant segments) tend to receive lower importance scores and thus lower budgets, concentrating resources where the approximation is most accurate.

Proposition C.4 (Cumulative Error under Sequential Compaction)**.** *Suppose TAM performs K sequential compaction steps. At step k, let ε_k denote the per-query attention output error of step k's compacted cache relative to the step-$(k-1)$ cache, and $\lambda_k \in [0, 1]$ the fraction of fresh tokens. Under a* linear error composition *model—where the attention output error from previously compacted tokens and fresh tokens combines additively, weighted by their respective fractions—the cumulative error relative to the original cache satisfies:*

$$\tilde{\varepsilon}_K \le \sum_{k=1}^{K} \varepsilon_k \cdot \prod_{j=k+1}^{K} (1 - \lambda_j). \tag{13}$$

When $\lambda_k \ge \lambda_{\min} > 0$ for all k, we have $\tilde{\varepsilon}_K \le \varepsilon_{\max}/\lambda_{\min}$, independent of K. In the periodic regime with interval P and target size t, $\lambda_k \approx P/(P + t)$.

Remark C.4 (Linear Composition Assumption)**.** The recurrence $\tilde{\varepsilon}_k \le \varepsilon_k + (1 - \lambda_k)\tilde{\varepsilon}_{k-1}$ models error propagation as a weighted average of fresh-step error and inherited error. In practice, errors pass through subsequent softmax operations, which are nonlinear; this can amplify errors (if compaction shifts attention toward high-error regions) or attenuate them (if fresh tokens dominate the softmax denominator). The linear model provides a tractable upper bound when per-step errors are small relative to the attention scores, which holds at moderate compression ratios (Table 6 confirms stable accuracy down to P=512). At extreme ratios or very frequent compaction (P=256), the linear bound may become loose. When the condition $\gamma > |\mathcal{P}|/t$ holds (see the remark following Proposition C.2), TAM reduces the per-step error ε_k relative to uniform compaction via adaptive allocation (Proposition C.1) and pivotal protection (Proposition C.2), directly tightening the cumulative bound. The compaction interval P further controls accumulation: larger P increases λ_k, limiting error propagation.

D Proofs of Theoretical Results

Proof of Proposition 4.1. **Part 1 (Mass deficit).** For eviction retaining subset S with $|S| = t$, the compacted mass is

$$\text{Mass}(\mathbf{q}; \mathbf{C}_k) = \sum_{j \in S} \exp(\mathbf{q}\mathbf{K}_j^\top / \sqrt{d}) = \text{Mass}(\mathbf{q}; \mathbf{K}) - \sum_{j \notin S} \exp(\mathbf{q}\mathbf{K}_j^\top / \sqrt{d}). \tag{14}$$

Since $S \neq [T]$, the second sum is strictly positive for all $\mathbf{q}$ (as $\exp(\cdot) > 0$), giving $\text{Mass}(\mathbf{q}; \mathbf{C}_k) = (1 - \mu(\mathbf{q})) \cdot \text{Mass}(\mathbf{q}; \mathbf{K}) < \text{Mass}(\mathbf{q}; \mathbf{K})$.

Part 2 (Concatenation effect). When concatenated with future keys $\mathbf{K}_f$, the softmax denominator is $\text{Mass}(\mathbf{q}; \mathbf{C}_k) + \text{Mass}(\mathbf{q}; \mathbf{K}_f)$. The compacted prefix's softmax weight is $(1 - \mu(\mathbf{q}))\text{Mass}(\mathbf{q}; \mathbf{K})/[(1 - \mu(\mathbf{q}))\text{Mass}(\mathbf{q}; \mathbf{K}) + \text{Mass}(\mathbf{q}; \mathbf{K}_f)]$, which is strictly less than the original $\text{Mass}(\mathbf{q}; \mathbf{K})/[\text{Mass}(\mathbf{q}; \mathbf{K}) + \text{Mass}(\mathbf{q}; \mathbf{K}_f)]$.

Part 3 (Output error). Let $\alpha_j = \exp(\mathbf{q}\mathbf{K}_j^\top / \sqrt{d})/Z$ be the original attention weights with $Z = \text{Mass}(\mathbf{q}; \mathbf{K})$, and $\hat{\alpha}_j = \alpha_j/(1 - \mu(\mathbf{q}))$ for $j \in S$ be the eviction weights. The output error (considering only the prefix block) is:

$$\|\hat{\mathbf{y}} - \mathbf{y}\| = \left\| \sum_{j \in S} \hat{\alpha}_j \mathbf{V}_j - \sum_{j=1}^{T} \alpha_j \mathbf{V}_j \right\| = \left\| \sum_{j \in S} \frac{\mu(\mathbf{q})}{1 - \mu(\mathbf{q})} \alpha_j \mathbf{V}_j - \sum_{j \notin S} \alpha_j \mathbf{V}_j \right\| \leq 2\mu(\mathbf{q}) \|\mathbf{V}\|_\infty, \tag{15}$$

where the last inequality uses $\sum_{j \in S} \alpha_j = 1 - \mu(\mathbf{q})$, $\sum_{j \notin S} \alpha_j = \mu(\mathbf{q})$, and triangle inequality. □

Proof of Proposition C.1. **Part 1 (KKT conditions).** We minimize $\mathcal{E}(\mathbf{t}) = \sum_i w_i \phi(n_i/t_i)$ subject to $\sum_i t_i = t$ and $t_i \geq 1$. Introducing Lagrange multiplier ν for the equality constraint, the stationarity condition at an interior point ($t_i > 1$) gives:

$$\frac{\partial \mathcal{E}}{\partial t_i} = -w_i \cdot \frac{n_i}{t_i^2} \cdot \phi'(n_i/t_i) = -\nu \quad \Longrightarrow \quad w_i \cdot \frac{n_i}{t_i^2} \cdot \phi'(n_i/t_i) = \nu \quad \forall i. \tag{16}$$

Since ϕ is convex and increasing, ϕ' is non-decreasing, and the second-order conditions are satisfied. Rearranging: $t_i^2 = w_i n_i \phi'(n_i/t_i)/\nu$, so $t_i^\star \propto \sqrt{w_i n_i \phi'(n_i/t_i^\star)}$, as stated.

Part 2 (Linear model). When $\phi(r) = r - 1$, we have $\phi'(r) = 1$ (constant) and $\phi''(r) = 0$. Returning to the stationarity condition: $w_i n_i / t_i^2 = \nu$, which gives $t_i = \sqrt{w_i n_i / \nu}$. Using the budget constraint $\sum_i t_i = t$:

$$t_i^\star = t \cdot \frac{\sqrt{w_i n_i}}{\sum_j \sqrt{w_j n_j}}. \tag{17}$$

Part 3 (Suboptimality of uniform). Under uniform allocation $t_i^{\text{unif}} = t \cdot n_i/N$ (where $N = \sum_j n_j$), the weighted error is $\mathcal{E}^{\text{unif}} = \sum_i w_i \phi(N/t)$, which depends only on the global ratio. Under optimal allocation, $\mathcal{E}^\star \leq \mathcal{E}^{\text{unif}}$ with equality iff the KKT conditions are satisfied by uniform t_i, which requires $w_i n_i/(n_i/N)^2 = \text{const}$, i.e., $w_i/n_i = \text{const}$. When importance density varies, the inequality is strict. □

Proof of Proposition C.2. Partition the token set into pivotal tokens $\mathcal{P}$ and non-pivotal tokens $\bar{\mathcal{P}} = [T] \setminus \mathcal{P}$. For any query $\mathbf{q}$, the ideal attention output decomposes as:

$$\mathbf{y}^* = \sum_{j \in \mathcal{P}} \alpha_j(\mathbf{q}) \mathbf{V}_j + \sum_{j \in \bar{\mathcal{P}}} \alpha_j(\mathbf{q}) \mathbf{V}_j = \gamma_{\mathbf{q}} \cdot \mathbf{y}_{\mathcal{P}}^* + (1 - \gamma_{\mathbf{q}}) \cdot \mathbf{y}_{\bar{\mathcal{P}}}^*, \tag{18}$$

where $\gamma_{\mathbf{q}} = \sum_{j \in \mathcal{P}} \alpha_j(\mathbf{q})$ is the attention mass on pivotal tokens for query $\mathbf{q}$, and $\mathbf{y}_{\mathcal{P}}^*$, $\mathbf{y}_{\bar{\mathcal{P}}}^*$ are the locally normalized outputs of each partition.

Pivotal tokens are retained exactly: $\mathbf{C}_k[\mathcal{P}, :] = \mathbf{K}[\mathcal{P}, :]$ and $\beta_j = 0$ for $j \in \mathcal{P}$ (Algorithm 1). Let $\hat{\gamma}_{\mathbf{q}}$ denote the attention mass on pivotal tokens under the compacted cache, and $\hat{\mathbf{y}}_{\bar{\mathcal{P}}}$ the locally normalized output of the non-pivotal compacted portion. The compacted output is:

$$\hat{\mathbf{y}} = \hat{\gamma}_{\mathbf{q}} \cdot \mathbf{y}_{\mathcal{P}}^* + (1 - \hat{\gamma}_{\mathbf{q}}) \cdot \hat{\mathbf{y}}_{\bar{\mathcal{P}}}. \tag{19}$$

Case 1: Exact mass matching. When total attention mass is matched ($\text{Mass}(\mathbf{q}; \mathbf{C}_k, \boldsymbol{\beta}) = \text{Mass}(\mathbf{q}; \mathbf{K})$), pivotal tokens' numerators are unchanged, so $\hat{\gamma}_{\mathbf{q}} = \gamma_{\mathbf{q}}$. The error simplifies to:

$$\|\hat{\mathbf{y}} - \mathbf{y}^*\| = (1 - \gamma_{\mathbf{q}}) \cdot \|\hat{\mathbf{y}}_{\bar{\mathcal{P}}} - \mathbf{y}^*_{\bar{\mathcal{P}}}\| \leq (1 - \gamma_{\mathbf{q}}) \cdot \varepsilon_{\text{rest}}. \tag{20}$$

Case 2: General bound with mass mismatch. Let $\Delta\gamma_{\mathbf{q}} = \hat{\gamma}_{\mathbf{q}} - \gamma_{\mathbf{q}}$. The error becomes:

$$\hat{\mathbf{y}} - \mathbf{y}^* = \Delta\gamma_{\mathbf{q}}(\mathbf{y}^*_{\mathcal{P}} - \hat{\mathbf{y}}_{\bar{\mathcal{P}}}) + (1 - \gamma_{\mathbf{q}})(\hat{\mathbf{y}}_{\bar{\mathcal{P}}} - \mathbf{y}^*_{\bar{\mathcal{P}}}). \tag{21}$$

Since $\mathbf{y}^*_{\mathcal{P}}$ and $\hat{\mathbf{y}}_{\bar{\mathcal{P}}}$ are both convex combinations of value vectors, $\|\mathbf{y}^*_{\mathcal{P}} - \hat{\mathbf{y}}_{\bar{\mathcal{P}}}\| \leq 2\|\mathbf{V}\|_\infty$. By triangle inequality:

$$\|\hat{\mathbf{y}} - \mathbf{y}^*\| \leq (1 - \gamma_{\mathbf{q}}) \cdot \varepsilon_{\text{rest}} + 2|\Delta\gamma_{\mathbf{q}}| \cdot \|\mathbf{V}\|_\infty. \tag{22}$$

Taking $\gamma = \min_{\mathbf{q}} \gamma_{\mathbf{q}}$ yields the stated bounds. AM's NNLS fitting enforces mass matching for the reference queries $\mathbf{Q}_{\text{ref}}$, so $\Delta\gamma_{\mathbf{q}} \approx 0$ for those queries and the clean bound applies. Note that $\varepsilon_{\text{rest}}$ depends on the reduced budget $t - |\mathcal{P}|$; pivotal protection yields a net error reduction whenever γ is large enough that $(1 - \gamma)\varepsilon_{\text{rest}} < \varepsilon_{\text{all}}$, where ε_{all} is the error of uniform compaction to size t. Under $\phi(r) = r - 1$, this holds when $\gamma > |\mathcal{P}|/t$. □

Proof of Proposition C.3. Let $\mathbf{X} \in \mathbb{R}^{n \times t}$ have rows $\mathbf{x}^{(i)}_{\text{ref}} = \text{softmax}(\mathbf{q}_i \mathbf{C}_k^\top / \sqrt{d} + \boldsymbol{\beta})$ and $\mathbf{Y} \in \mathbb{R}^{n \times d}$ have rows $\mathbf{y}_i$ (the target attention outputs from the original cache). The OLS solution $\mathbf{C}^\star_v = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}$ minimizes $\|\mathbf{X}\mathbf{C}_v - \mathbf{Y}\|_F^2$.

For a new query with attention pattern $\mathbf{x}_{\mathbf{q}} \in \mathbb{R}^{1 \times t}$, let $P = \mathbf{X}^\top (\mathbf{X}\mathbf{X}^\top)^{-1} \mathbf{X} \in \mathbb{R}^{t \times t}$ denote the projection onto the row space of $\mathbf{X}$. Decompose $\mathbf{x}_{\mathbf{q}}^\top = P\,\mathbf{x}_{\mathbf{q}}^\top + (I - P)\,\mathbf{x}_{\mathbf{q}}^\top$, and define the row vectors $\mathbf{x}^{\|}_{\mathbf{q}} = (P\,\mathbf{x}_{\mathbf{q}}^\top)^\top$ and $\mathbf{x}^{\perp}_{\mathbf{q}} = ((I - P)\,\mathbf{x}_{\mathbf{q}}^\top)^\top$, so that $\mathbf{x}_{\mathbf{q}} = \mathbf{x}^{\|}_{\mathbf{q}} + \mathbf{x}^{\perp}_{\mathbf{q}}$. The compacted output is $\hat{\mathbf{y}} = \mathbf{x}_{\mathbf{q}} \mathbf{C}^\star_v = \mathbf{x}^{\|}_{\mathbf{q}} \mathbf{C}^\star_v + \mathbf{x}^{\perp}_{\mathbf{q}} \mathbf{C}^\star_v$.

By the triangle inequality:

$$\|\hat{\mathbf{y}} - \mathbf{y}^*\| \leq \|\mathbf{x}^{\|}_{\mathbf{q}} \mathbf{C}^\star_v - \mathbf{y}^*_{\|}\| + \|\mathbf{x}^{\perp}_{\mathbf{q}} \mathbf{C}^\star_v\| \leq \|\mathbf{x}^{\|}_{\mathbf{q}} \mathbf{C}^\star_v - \mathbf{y}^*_{\|}\| + \|\mathbf{x}^{\perp}_{\mathbf{q}}\| \cdot \|\mathbf{C}^\star_v\|_{\text{op}}, \tag{23}$$

where $\|\mathbf{C}^\star_v\|_{\text{op}}$ is the operator (spectral) norm. When $\mathbf{x}^{\perp}_{\mathbf{q}} = \mathbf{0}$, the second term vanishes. For $\mathbf{x}^{\|}_{\mathbf{q}}$: since it lies in the row space of $\mathbf{X}$, we can write $\mathbf{x}^{\|}_{\mathbf{q}} = \boldsymbol{\lambda}^\top \mathbf{X}$ for some $\boldsymbol{\lambda} \in \mathbb{R}^n$, so $\mathbf{x}^{\|}_{\mathbf{q}} \mathbf{C}^\star_v = \boldsymbol{\lambda}^\top \mathbf{X}\mathbf{C}^\star_v$. The residual $\mathbf{X}\mathbf{C}^\star_v - \mathbf{Y}$ is the OLS fitting error, so $\mathbf{x}^{\|}_{\mathbf{q}} \mathbf{C}^\star_v = \boldsymbol{\lambda}^\top (\mathbf{Y} + (\mathbf{X}\mathbf{C}^\star_v - \mathbf{Y}))$, yielding the stated bound. □

Proof of Proposition C.4. We proceed by induction under the linear error composition model. At step 1, $\tilde{\varepsilon}_1 = \varepsilon_1$ (the error of the first compaction relative to the original cache). Assume $\tilde{\varepsilon}_{k-1}$ bounds the step-$(k-1)$ error relative to the original.

At step k, the cache consists of two parts: the step-$(k-1)$ compacted prefix (with error $\tilde{\varepsilon}_{k-1}$) and fresh tokens generated since step $k-1$ (with no compaction error). The fraction of fresh tokens is λ_k. Under the linear composition model, the attention output can be decomposed as a weighted sum of the contributions from previously-compacted tokens and fresh tokens; since softmax is approximately linear for small perturbations when no single token dominates, the inherited error is attenuated by the previously-compacted fraction. After step-k compaction with per-step error ε_k:

$$\tilde{\varepsilon}_k \leq \varepsilon_k + (1 - \lambda_k) \cdot \tilde{\varepsilon}_{k-1}, \tag{24}$$

where the $(1 - \lambda_k)$ factor reflects that only the previously-compacted fraction propagates old errors, while fresh tokens contribute no inherited error. Unrolling the recurrence:

$$\tilde{\varepsilon}_K \leq \sum_{k=1}^{K} \varepsilon_k \cdot \prod_{j=k+1}^{K} (1 - \lambda_j). \tag{25}$$

When $\lambda_k \geq \lambda_{\min}$ and $\varepsilon_k \leq \varepsilon_{\max}$ for all k:

$$\tilde{\varepsilon}_K \leq \varepsilon_{\max} \sum_{k=1}^{K} (1 - \lambda_{\min})^{K-k} = \varepsilon_{\max} \cdot \frac{1 - (1 - \lambda_{\min})^K}{\lambda_{\min}} \leq \frac{\varepsilon_{\max}}{\lambda_{\min}}. \tag{26}$$

In the periodic regime with interval P and target size t, the cache at compaction has $\approx t + P$ tokens of which P are fresh, giving $\lambda_k \approx P/(P + t)$.

Connection to TAM. Under TAM, the per-step error ε_k is reduced relative to uniform compaction via pivotal protection (Proposition C.2) and optimal allocation (Proposition C.1). When the condition $\gamma > |\mathcal{P}|/t$ holds (see the remark following Proposition C.2), TAM's per-step error satisfies $\varepsilon_k^{\text{TAM}} < \varepsilon_k^{\text{uniform}}$, yielding a tighter cumulative bound. □

E Full Experimental Results

E.1 Full Results: Accuracy vs. Target Size

Table 7 reports accuracy and post-compaction memory across all methods and target compaction ratios on both benchmarks, corresponding to Figure 2a and Figure 2b in the main text.

Table 7: Full results on **AIME 2024** (Qwen3-4B, 30 problems): accuracy (%) / post-compaction memory (GB) at each target compaction ratio.

Method	**Target Compaction Ratio**		
	0.05	0.10	0.20
No Compaction	63.3% / 9.2 GB		
Eviction (Selector Window)	33.3 / 3.0	46.7 / 4.1	53.3 / 5.7
AM + Repeat	46.7 / 3.1	53.3 / 4.2	56.7 / 5.8
PAM (Uniform)	50.0 / 3.0	56.7 / 4.0	60.0 / 5.6
TAM (Ours)	**53.3** / 3.0	**60.0** / 4.0	**63.3** / 5.6
TAM (Periodic, P=1024)	50.0 / <u>2.4</u>	56.7 / <u>3.2</u>	60.0 / <u>4.5</u>

Table 8: Full results on **MATH-500** (Qwen3-4B): accuracy (%) / post-compaction memory (GB) at each target compaction ratio.

Method	**Target Compaction Ratio**		
	0.05	0.10	0.20
No Compaction	71.2% / 8.8 GB		
Eviction (Selector Window)	38.4 / 2.9	52.4 / 3.9	61.8 / 5.4
AM + Repeat	49.2 / 3.0	61.0 / 4.0	66.2 / 5.5
PAM (Uniform)	52.6 / 2.9	64.6 / 3.8	68.4 / 5.3
TAM (Ours)	**56.0** / 2.9	**67.8** / 3.8	**70.2** / 5.3
TAM (Periodic, P=1024)	53.2 / <u>2.3</u>	65.4 / <u>3.1</u>	68.0 / <u>4.3</u>

E.2 Selector Window Size R

Table 9 reports accuracy versus selector window size R under TAM. Larger R provides more reference queries but increases extraction cost. We find R=64 to offer a good trade-off.

Table 9: Ablation on selector window size R (TAM, target ratio 0.1, AIME 2024).

Window R	**Acc (%)**	**Extract Time (s)**
32	56.7	0.4
64 (default)	**60.0**	0.8
128	60.0	1.5