Free(): Your LLM is Leaking Memory

Anonymous ACL submission

Abstract

Standard LLMs operate as "malloc-only" engines. They continuously accumulate tokens in their reasoning chains but lack the `free()` operation to release information that is no longer needed. **This leads to a "cognitive memory leak"**: the context window—acting as the model's logical RAM—becomes cluttered with redundancy, ranging from dead-end paths to transient verification steps. This noise **distracts** the attention, causing the model to struggle with maintaining a clear chain of thought.

In this paper, we introduce **`Free()LM`**, an architecture that integrates an intrinsic `free()` function to actively manage this cognitive memory. We equip the Vanilla model with a lightweight, trainable **Free-Module**. Activated at regular intervals during generation, this module outputs a structured command (indicating a `prefix` and `suffix`) to precisely locate redundant reasoning steps. By surgically excising redundancy from the context, `Free()LM` transforms the linear CoT into a dynamic, managed workspace.

Our experiments confirm that the **Free-Module** significantly enhances reasoning. Across six long-reasoning benchmarks, **`Free()LM`** improves Qwen3-8B and Qwen3-30B-A3B by an average of **4.4%**. On the massive Qwen3-235B-A22B, it yields a **11% relative gain** (16.1% → 18.0%) on Human's Last Exam (HLE). Notably, on complex cases requiring 70k+ thinking tokens, it boosts Qwen3-235B-A22B's accuracy from **0% to 28%**. Crucially, this performance comes at a lower cost: it halves the KV cache memory usage on HLE (6.14GB to 3.34GB per sample), proving that the key to infinite reasoning is not just remembering everything, but knowing when to forget.

1 Introduction

LLMs have become so powerful that they are even envisioned as the kernel of a new operating system (Karpathy, 2023). However, viewed through a software engineering lens, this "system" exhibits a fatal design flaw: it operates as a **"malloc-only" engine**. As illustrated in Figure 1 (Left), standard models continuously allocate memory for new tokens but lack the essential `free()` operation to discard information that is obsolete. We pour resources into expanding the context window—effectively upgrading the RAM—yet we never teach the model how to clean up. Consequently, with greater context comes not greater power, but greater noise.

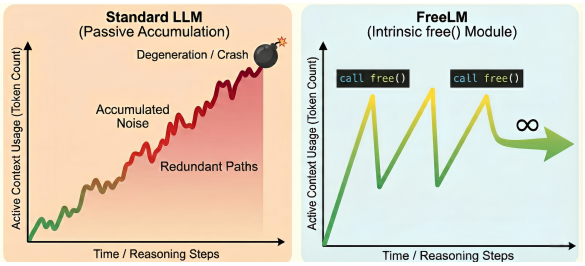

Figure 1: **The "Malloc-only" Engine vs. `Free()LM`. Left:** Standard LLMs passively accumulate tokens, and cause the reasoning process to eventually "crash" (degenerate) once the context is polluted. **Right:** `Free()LM` integrates an intrinsic `free()` operation. By periodically identifying and removing redundant steps (via `call free()`), it actively maintains a compact, managed workspace, enabling stable and infinite reasoning.

We empirically validated this "Cognitive Memory Leak" on the AIME24 (Zhang and Math-AI, 2024) and AIME25 (Zhang and Math-AI, 2025) benchmarks using Qwen3-8B (Yang et al., 2025). From 480 sampled reasoning paths, we identified 31 instances truncated by the standard 32k context limit. Through a case-by-case analysis, we found that 26 of these instances (84%) had already collapsed into catastrophic degeneration[1], trapped in loops of repetitive mistakes. Crucially, extending the context window to 128k failed to rescue these cases; the model simply continued to repeat the

[1]This is a conservative estimate; lowering the decoding temperature exacerbates degeneration.

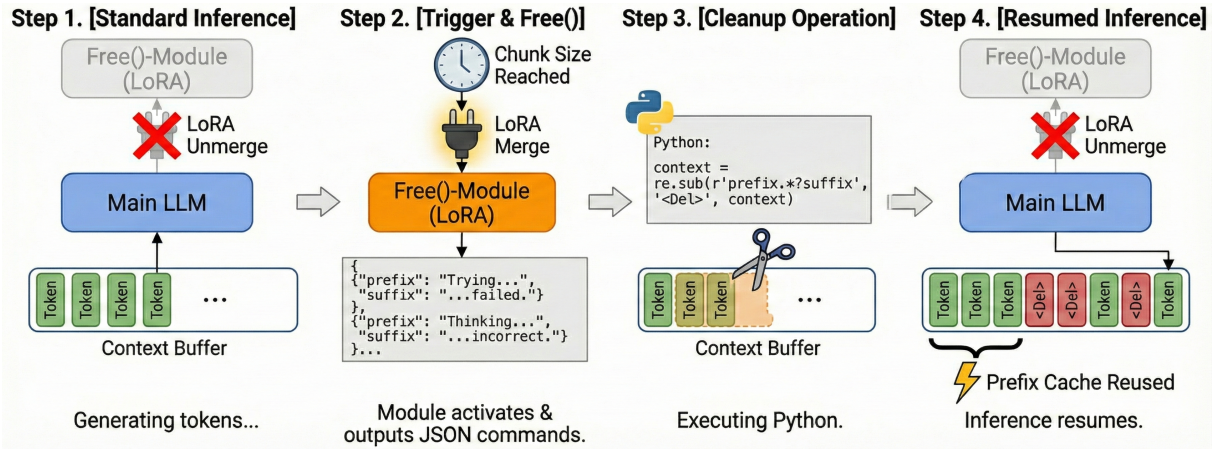

Figure 2: **The `Free()LM` Architecture & Inference Flow.** The process alternates between generation and clean-up phases, enabling infinite reasoning within a fixed memory budget.

same error patterns in the expanded space. In other words, once the context—the model's "cognitive RAM"—is polluted, the reasoning process crashes, and the model has **no choice but to restart**.

To break this cycle, we introduce **`Free()LM`**, an architecture that finally equips LLMs with a `free()` function. Instead of forcing a hard restart when the context is polluted, `Free()LM` actively cleans it up on the fly, creating the dynamic memory profile shown in Figure 1 (Right). We add a lightweight, trainable **Free-Module** (via LoRA) that works directly within the generation process. Activated at regular intervals, this module analyzes the current reasoning trace and outputs a structured command (specifying a `prefix` and `suffix`) to excise redundant steps—ranging from erroneous dead ends to successful but verbose verification traces that are no longer needed. This operation transforms the context from a messy, linear log into an efficient, managed workspace.

Our empirical results validate that this approach works. While `Free()LM` delivers a solid 4.4% gain on standard benchmarks, its true value lies in the extremes. On the challenging HLE, it boosts the performance of Qwen3-235B-A22B-Thinking-2507 (hereafter Qwen3-235B-A22B) by 11% relative to the Vanilla baseline. The gap is most visible on complex cases requiring 70k+ thinking tokens: where the standard model, Qwen-235B-A22B, fails completely (0%), `Free()LM` surges to a remarkable 28%. Crucially, we achieve this not by adding resources, but by removing waste—halving the KV memory cost (6.14GB $\rightarrow$ 3.34GB). This confirms our core proposition: the key to infinite reasoning is not just remembering everything, but knowing what to forget.

2 Method

To mitigate context pollution, we introduce **`Free()LM`**, an architecture augments the standard LLM backbone with a lightweight, plug-and-play **Free-Module**. As illustrated in Figure 2, the system operates by alternating between two distinct phases: **Generation** (acting as `malloc()`) and **Clean-Up** (acting as `free()`). During Generation, the model produces reasoning steps autoregressively; during Clean-Up, the Free-Module is activated to identify and excise redundant information. This mechanism ensures that the reasoning workspace remains compact and managed, regardless of trajectory length.

2.1 Inference with `Free()`

The `free()` operation is implemented as an intervention within the generation loop, controlled by a new decoding parameter.

Triggering: We introduce a decoding hyperparameter, the Clean-Up size (L_{clean}). The system acts as a monitor during standard inference. For every L_{clean} tokens generated, the process interrupts the generation and merge the Free-Module into the main LLM.

The Free-Module: As shown in Step 2 of Figure 2, upon activation, this module scans the previously generated context and outputs a structured command in JSON:

```
[{"prefix": "...", "suffix": "..."},...]
```

The `prefix` and `suffix` act as unique string anchors, precisely defining the span [start, end] containing redundant information—whether it is a dead-end path or a transient verification step. Such a design enables the Free-Module to efficiently delete long sequences with very few output tokens.

Clean-Up: An external Python executor parses the JSON command and physically removes the target span from the context buffer. As illustrated in Step 3 of Figure 2, the executor applies the free() operations using standard regular expressions:

```
context= re.sub(prefix + r'.*?'
       + suffix,'<Del>', context)
```

Following this excision, the system unmerges the LoRA and seamlessly resumes standard inference.

Resumed Inference: To resume generation with the refined context C_{new}, we identify two implementation strategies:

1. **Re-prefilling:** We reuse the KV cache corresponding to the unchanged prefix of C_{new} and strictly re-prefill the altered suffix.

2. **KV Cache Pruning:** We directly excise the deleted memory blocks. To address the positional shift of subsequent tokens, we rotate their cached Key-Value states to re-align with the correct position indices. (Ma et al.)

Our pilot experiments on smaller models indicate that both strategies yield nearly identical performance. However, since **Strategy 2** is not natively supported by standard serving frameworks like vLLM (Kwon et al., 2023), we adopt **Strategy 1** for our main experiments to ensure compatibility and efficiency.

2.2 Training: Learning to Forget

Given the concept of active context management, a natural starting point is to explore In-Context Learning (ICL) solutions: asking the model itself to conduct self-correction or employing a strong external LLM to identify redundancy. Unfortunately, our preliminary experiments (detailed in Table 1) reveal the severe limitations of this approach. Even with extensive prompt optimization and utilizing powerful models like Gemini-2.5-Pro (Comanici et al., 2025) as experts, the performance gains on Qwen3-8B were marginal ($\sim$ 1%). This finding suggests that effectively executing the `free()` operation—distinguishing necessary historical context from obsolete noise—is a complex capability that must be acquired through explicit training.

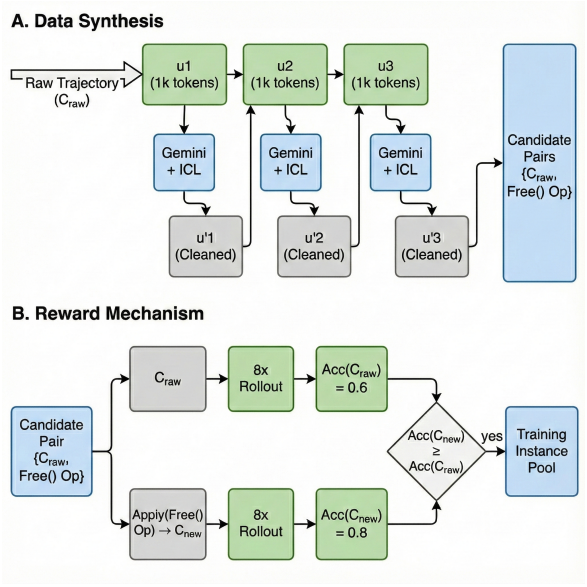

Figure 3: The Training Data Pipeline.

To address the lack of ground-truth labels, we propose a data pipeline centered on a **sophisticated reward mechanism**. Specifically, we first synthesize a large pool of candidate `free()` operations via ICL solutions, and subsequently filter for high-quality instances using the reward mechanism. The overall pipeline is illustrated in Figure 3.

Data Synthesis We randomly select 1,000 trajectories from DeepMath-103k (He et al., 2025) and synthesize supervision signals using Gemini-2.5-Pro. Crucially, to mimic the run-time behavior, we adopt a **sequential cleaning** strategy. We segment each raw trajectory T into sequential 1k-token chunks $\{u_1, u_2, \ldots, u_n\}$. The cleaning process is performed iteratively: when the oracle cleans the current chunk u_k, it is conditioned on the **already cleaned** history prefix H'_{k-1} (composed of previous cleaned chunks) rather than the original raw context. This dependency chain ensures that the training data reflects the fragmented memory states the model will actually encounter during inference. This pipeline yielded $\sim$8,000 candidate instances.

Reward Mechanism To filter the synthesized candidates, we employ a rigorous rejection sampling strategy. We treat the dataset as pairs of $\{C_{\text{raw}}, \mathcal{O}\}$, where C_{raw} is the original context and $\mathcal{O}$ denotes the candidate `free()` operation.

For each pair, we execute $K = 8$ independent reasoning rollouts on both the original context C_{raw}

and the cleaned context $C_{\text{new}} = \text{Apply}(C_{\text{raw}}, \mathcal{O})$. A candidate operation is retained if and only if the cleaning preserves or improves the accuracy:

$$\text{Acc}(C_{\text{new}}) \geq \text{Acc}(C_{\text{raw}})$$

This strict criterion ensures that the module learns to excise noise without reducing the probability of reaching the correct solution. Consequently, this verification process distilled the dataset down to 6,648 high-quality training instances.

3 Experiments

We evaluate `Free()LM` across a diverse set of benchmarks to verify its effectiveness, generalization capability, and efficiency. Our experiments cover models ranging from 8B to 235B parameters.

3.1 Settings

Backbone Models. We evaluate `Free()LM` across three model scales: Qwen3-8B, Qwen3-30B-A3B-Thinking-2507, and Qwen3-235B-A22B-Thinking-2507 (hereafter denoted as **Qwen3-8B**, **Qwen3-30B-A3B**, and **Qwen3-235B-A22B** for brevity). We set the context window to 32k for Qwen3-8B and Qwen3-30B-A3B, as most reasoning paths fit within this limit. For the largest model, Qwen3-235B-A22B, we allocate a 128k window for the **Vanilla** baseline to accommodate its extensive generation length, while restricting `Free()LM` to 64k, leveraging its ability to actively free up memory during inference.

Compared Methods. We compare our `Free()LM` with three types of baselines:

- **Vanilla**: The standard backbone model, **Qwen3-8B**, **Qwen3-30B-A3B**, and **Qwen3-235B-A22B** (Yang et al., 2025) without any context management or compression techniques.
- **Heuristic Compression**: Since KV cache compression naturally results in a shorter context, we adapt **H2O** (Zhang et al., 2023) and ThinkCleary (**TC**) (Choi et al., 2025) as heuristic baselines. We apply these methods dynamically during reasoning to prune tokens based on their accumulated attention scores. Note that due to their incompatibility with the vLLM (Kwon et al., 2023) serving framework, we evaluate them exclusively on Qwen3-8B.
- **ICL Methods**: the ICL solutions discussed in Section 2.2. We instruct both the backbone models (denoted as **No Train**) and the SOTA **Gemini**-2.5-Pro (Comanici et al., 2025) to execute `free()` operations via a sophisticatedly optimized prompt detailed in Appendix A.1.

Benchmarks. To comprehensively evaluate `Free()LM`, we carefully design our benchmark selection to stress-test two critical hypotheses: (1) whether active context management mitigates the "Cognitive Memory Leak" on long reasoning chains, and (2) whether the `free()` operation preserves performance on short reasoning tasks that trigger fewer Clean-Up operations.

To evaluate long-context reasoning, we employ a suite of challenging benchmarks: AIME2425 (Zhang and Math-AI, 2024, 2025),[2] BrUMO25 (Balunović et al., 2025a), HMMT (Balunović et al., 2025b), BeyondAIME (ByteDance-Seed, 2025), Human Last Exam (HLE) [3] (Phan et al., 2025), and IMOAnswerBench (Luong et al., 2025). These benchmarks were selected because they are particularly prone to the failure modes targeted by `Free()LM`. Specifically, they require complex, multi-step reasoning that often results in trajectories of 10k–20k+ tokens, which frequently trigger context pollution and catastrophic degeneration in standard models. Furthermore, the recency of these datasets, such as BeyondAIME (June 2025) and IMOAnswerBench (November 2025), minimizes the risk of data contamination and ensures a robust evaluation of true reasoning capabilities.

To ensure that `Free()LM` retains its general reasoning capabilities, we evaluate it on several short-reasoning benchmarks: BBH (Suzgun et al., 2022), MMLU-Pro (Wang et al., 2024), MMLU-STEM (Hendrycks et al., 2020), and GPQA-Diamond (Rein et al., 2024). These tasks, characterized by shorter trajectories, serve to verify that the introduction of the `free()` operation maintains performance even when memory management is less frequently required.

Implementation Details. For mathematical reasoning tasks, we standardize outputs using the prompt: "`Please reason step by step, and put your final answer within \boxed{}.`" Correctness is evaluated by extracting the answer

[2] AIME24 (Zhang and Math-AI, 2024) and AIME25 (Zhang and Math-AI, 2025) are combined into a single dataset, hereafter AIME2425.

[3] We use text-only subset of HLE.

Table 1: Performance of Qwen3 models. We report pass@1 (p@1) performance computed over 8 rollouts, along with the average number of response tokens (#Token).

Model	Setting	AIME2425		BrUMO25		HMMT		BeyondAIME		HLE		IMOAnswer		Average	
		p@1 ↑	#Token ↓ (Δ)	p@1 ↑	#Token ↓ (Δ)	p@1 ↑	#Token ↓ (Δ)	p@1 ↑	#Token ↓ (Δ)	p@1 ↑	#Token ↓ (Δ)	p@1 ↑	#Token ↓ (Δ)	p@1 ↑	#Token ↓ (Δ)
Qwen3 -8B	Vanilla	71.67	17.6k	69.58	15.8k	38.75	19.4k	42.38	19.5k	4.59	13.3k	38.50	19.4k	44.24	17.5k
	H2O	60.00	17.8k (+1%)	70.00	18.3k (+16%)	46.67	21.5k (+11%)	35.00	20.5k (+5%)	4.39	15.3k (+15%)	36.25	20.7k (+7%)	42.05	19.0k (+9%)
	TC	51.67	20.3k (+16%)	66.67	16.6k (+5%)	26.67	20.5k (+6%)	42.00	19.7k (+1%)	4.96	15.3k (+14%)	34.00	20.3k (+4%)	37.66	18.8k (+7%)
	No Train	73.33	15.3k (-13%)	70.83	16.4k (+4%)	42.50	19.9k (+3%)	43.38	20.6k (+6%)	**5.89**	12.3k (-8%)	38.75	17.4k (-10%)	45.78	17.0k (-3%)
	Gemini	71.67	13.7k (-22%)	68.75	15.2k (-4%)	44.17	18.4k (-5%)	43.63	17.6k (-10%)	5.10	11.8k (-11%)	40.00	17.8k (-8%)	45.55	15.8k (-10%)
	`Free()LM`	**75.00**	13.0k (-26%)	**72.50**	13.7k (-13%)	**49.58**	16.4k (-15%)	**45.88**	15.4k (-21%)	5.38	9.9k (-26%)	**40.50**	14.3k (-26%)	**48.14**	13.8k (-21%)
Qwen3 -30B	Vanilla	83.33	14.9k	82.92	15.1k	60.83	21.0k	56.50	22.5k	8.99	13.9k	52.25	21.1k	57.47	18.1k
	No Train	85.21	15.9k (+7%)	**86.25**	17.1k (+13%)	63.33	20.4k (-3%)	59.25	20.8k (-8%)	9.78	12.7k (-9%)	53.25	19.4k (-8%)	59.51	17.7k (-2%)
	Gemini	**87.92**	15.4k (+4%)	84.17	15.9k (+6%)	65.42	19.0k (-10%)	59.75	19.7k (-12%)	9.59	12.8k (-8%)	54.00	20.4k (-3%)	60.14	17.2k (-5%)
	`Free()LM`	**87.92**	14.5k (-2%)	85.42	14.0k (-8%)	**70.42**	19.1k (-9%)	**62.25**	18.2k (-19%)	**9.82**	11.3k (-19%)	**58.00**	18.1k (-14%)	**62.30**	15.9k (-12%)
Qwen3 -235B	Vanilla	92.29	21.7k	91.67	19.9k	**85.00**	29.1k	69.00	31.0k	16.13	22.5k	61.00	32.3k	69.18	26.1k
	No Train	90.63	16.2k (-26%)	92.08	17.1k (-14%)	80.00	24.4k (-16%)	68.63	22.9k (-26%)	15.20	18.2k (-19%)	58.25	24.4k (-25%)	67.46	20.5k (-21%)
	Gemini	**93.54**	18.5k (-15%)	93.75	17.7k (-11%)	84.17	25.6k (-12%)	**70.38**	26.5k (-14%)	16.59	18.7k (-17%)	62.25	27.6k (-15%)	70.11	22.4k (-14%)
	`Free()LM`	93.13	16.4k (-24%)	**94.58**	16.4k (-18%)	84.58	21.7k (-26%)	69.75	21.3k (-31%)	**18.03**	15.8k (-30%)	**62.75**	24.2k (-25%)	**70.47**	19.3k (-26%)

Table 2: Performance of `Free()LM` (Qwen3-8B Backbone) across short reasoning datasets.

Method	BBH	MMLU-Pro	MMLU-STEM	GPQA
Vanilla Model	82.86	75.57	92.29	59.41
`Free()LM`	82.89	75.58	92.55	**61.36**

from "`\boxed{}`" and matching it against the ground truth, following the evaluation pipeline (He et al., 2025). We report three key metrics: pass@1, the number of response tokens (#Token), and the response reduction ratio (Δ) relative to the vanilla model. For all experiments, we employ sampling-based decoding with temperature=0.7, top_k=20, and top_p=0.95. To balance computational cost with statistical reliability, we configure the number of rollouts based on dataset size: 8 rollouts for AIME2425, BrUMO25, HMMT, and BeyondAIME (which have limited questions), and 1 rollout for the larger HLE and IMOAnswerBench datasets. For `Free()LM`, the Clean-Up size L_{clean} is set to 5000 with a maximum of 50 iterations.

3.2 Main Results

We present the evaluation results on long-reasoning (Table 1) and short-reasoning tasks (Table 2). Recent advances in Test-Time Scaling suggest a rigid exchange rate: higher intelligence mandates higher inference costs (i.e., longer CoT). However, our results defy this conventional law. As shown in Table 1, `Free()LM` achieves a dual victory, simultaneously scaling up performance while reducing the inference-time memory footprint.

Higher Accuracy with Deep Cleaning: The most critical finding is the substantial boost in reasoning capability. On the Qwen3-8B benchmark, `Free()LM` achieves an average Pass@1 of **48.14%**, outperforming the vanilla inference (44.24%) by a remarkable margin of **+3.9%**, and surpassing the Gemini-2.5-Pro baseline by **+2.2%**. Crucially, this performance scaling is achieved not by generating *more*, but by retaining *less*. We observe a distinct "Deep Cleaning" phenomenon where `Free()LM` yields a significantly shorter average response length (**13.8k**) compared to the Gemini-2.5-Pro baseline (15.8k).

Our qualitative investigation reveals the decisive factor: while Gemini often mistakenly removes useful clues, forcing the backbone to regenerate them, `Free()LM` targets **true redundancy**, with virtually no regeneration observed. This contrast proves the power of our Reward Mechanism: we successfully trained a module that knows exactly what to delete, with a level of precision that even SOTA models often struggle to achieve.

The Failure of Heuristic Compression: The results for H2O and TC are puzzling. Not only did they fail to improve accuracy, but surprisingly, they also failed to reduce the response length. A close inspection of the logs reveals the reason: catastrophic degeneration. In these cases, the model gets stuck in repetitive loops, continuing to generate until hitting the maximum output token limit. Ultimately, instead of making the model "more intelligent" via cleanup, they disrupt the context, rendering the reasoning process significantly more prone to crashing.

Scalability: On the massive Qwen3-235B-A22B, one might notice that gains on standard benchmarks (AIME, BrUMO) appear modest. We attribute this to a performance ceiling: the model with vanilla inference manner already solves $> 90\%$ of these problems, leaving little room for improvement. However, `Free()LM` demonstrates its true value on challenging tasks like HLE and IMOAnswer,

Table 3: **Cross-Model Generalization.** The 8B Free-Module successfully enhances the performance of the 235B and 685B models on IMOAnswer, measured by Pass@1 and the reduction ratio of response tokens.

Inference Setup	Pass@1	Reduction
Qwen3-235B-A22B	61.00%	-
+ Free-Module (8B)	**62.50%**	**21.34%**
DeepSeek-V3.2-Speciale†	71.00%	-
+ Free-Module (8B)	**75.50%**	**45.99%**

† Despite utilizing the official model checkpoints, our evaluation yields results lagging behind the technical report by ∼15%. The exact cause of this discrepancy remains unidentified due to limited investigation time. So the significant performance advantage of `Free()LM` over this baseline shown here should be **interpreted with caution**, as our results likely under-represent its true capability.

Table 4: **Efficiency Comparison on HLE Benchmark.** We report the average per-sample latency and KV cache memory usage for Qwen3-235B-A22B.

Metric	Vanilla	`Free()LM`	Δ
Avg. Latency (s)	353.2	552.3	+56.4%
Avg. KV (GB)	6.14	**3.34**	**-45.6%**

it boosts accuracy significantly (e.g., +1.9% on HLE) while reducing context length by a massive **27.5%**. This sends a clear message: **bigger is not always better if the context is messy**. To perform at their peak, these giants not only need more memory—but also the ability to `free()` it.

Short-Reasoning Tasks: A common concern is whether "forgetting" hurts general capabilities. Table 2 allays this fear, showing `Free()LM` maintains parity across broad benchmarks. The reason is straightforward: we freeze the backbone. On these tasks, the Free-Module is triggered significantly less often. As a result, `Free()LM` closely tracks the performance of the vanilla model.

3.3 Cross-Model Generalization

Following the superior performance of `Free()LM`, we investigate a critical question: is the capability of a trained Free-Module model-specific, or does it generalize across different architectures? To explore this, we first apply the Free-Module trained on Qwen3-8B to perform the `free()` operation for the much larger Qwen3-235B-A22B on IMOAnswer. Results in Table 3 demonstrate that the 8B Free-Module achieves performance gains (+1.5%) comparable to those of the 235B Free-Module (+1.75% as previously shown in Table 1).

However, **a critical question remained**: is this generalization capability merely a byproduct of the shared model family? To challenge this hypothesis, we introduced a completely different architecture—**DeepSeek-V3.2-Speciale** (DeepSeek-AI et al., 2025)—into our evaluation. The results addressed this concern: even on this "alien" architecture, the Qwen-backboned 8B Free-Module improved Pass@1 by **4.5%** while simultaneously slashing response tokens by **45.99%**. This substantial improvement across distinct architectures suggests that the 8B-module has acquired a *universal* context-cleaning capability. This cross-model generalization inspires a plug-and-play deployment strategy: serving the Free-Module as a **"Universal Context Clean-Up Service."** In this paradigm, any backbone model can simply invoke this service upon reaching a trigger condition, clean-up its context, and resume inference without requiring model-specific adaptation.

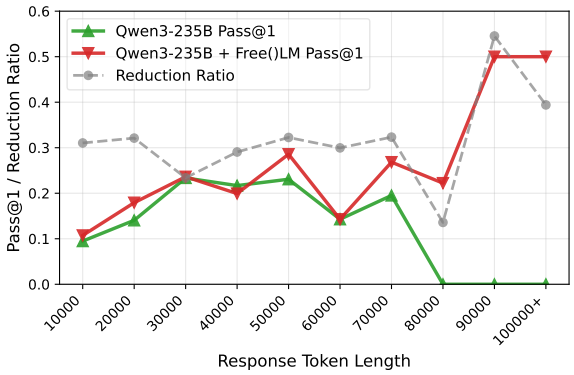

Figure 4: **Performance vs. Reasoning Length on HLE.** While standard Qwen3-235B-A22B suffers from "Intelligence Degradation" on trajectories longer than 80k tokens, `Free()LM` exhibits a striking rebound in accuracy. On these cases, the Free-Module reduces the context length by ∼45%, effectively mitigating context pollution.

3.4 Analysis on Reasoning Length

Figure 4 breaks down the performance on HLE. To isolate the impact of reasoning depth, we categorize test instances based on the token count of the response generated by the vanilla Qwen3-235B-A22B model. The results reveal a stark contrast:

Vanilla models suffer from "Intelligence Degradation." For manageable lengths, the vanilla model performs well. However, as trajectories extend beyond 80k tokens, the accumulated noise triggers a catastrophic degradation. This catastrophic degradation effectively nullifies the model's reasoning, resulting in a complete loss of accuracy.

Free()LM recovers the "Lost" Intelligence. In these same deep waters (>80k tokens), Free()LM exhibits a striking rebound, with accuracy surging back to ∼50%. Although the limited sample size in this tail warns against claiming a definitive "intelligence boost," the ability to *sustain* reasoning capability is undeniable.

The grey dashed line explains this recovery. On these ultra-long paths, the Free-Module aggressively compresses the context by 40%–50%. This operation successfully brings a massive 100k+ trajectory back down to the 40k–70k range—a "sweet spot" where the Qwen3-235B-A22B backbone operates most comfortably. We may simply conclude the result as: **with greater context comes a greater necessity for Free()LM.**

3.5 System Performance

Finally, we evaluate the engineering impact on Qwen3-235B-A22B using the HLE benchmark. We deploy the model across two 8×H20 nodes using Tensor Parallelism (TP=16). The serving backend is vLLM (v0.8.5) on CUDA 12.6, configured with FlashAttention-2 and BF16 precision.

We start with the cost: Free()LM incurs a 56% increase in latency per sample. This overhead primarily stems from three sources: (1) the time spent decoding free() commands, (2) the re-prefilling latency incurred after executing a deletion, and (3) the regeneration of information if the model occasionally over-prunes.

However, the upside is substantial: we achieve a **45% reduction** in KV cache usage (6.14 GB → 3.34 GB). In real-world serving, where memory bandwidth is often the bottleneck, this saving is critical. Furthermore, the current latency is not a hard limit. By adopting **Strategy 2** (KV Cache Pruning) to eliminate the re-prefilling step mentioned in (2), we estimate the overhead could be cut down to ∼20%, although we have not yet implemented this strategy in vLLM.

3.6 Case Study

We qualitatively compare Free()LM and an ICL-based Gemini baseline using an AIME2425 example (Figure 5). Free()LM (left) demonstrates precise deletion: when the model begins redundant self-correction (denoted by red), the Free-Module removes it, allowing inference to resume (denoted by green) without re-generating the pruned content. Conversely, Gemini (right) misjudges essential coordinate-setup as redundant. Its immediate regeneration (denoted by blue) confirms the deletion was erroneous, as the backbone still required that information. This "delete-then-regenerate" cycle highlights Gemini's inability to distinguish truly disposable context, resulting in wasted computation. Thus, it explains why Free()LM outperforms the Gemini-based Free-Module in Table 1.

4 Related Work

Free()LM tackles context accumulation—a challenge where unbounded intermediate thoughts crowd out information needed for subsequent reasoning. We situate our approach at the intersection of KV cache compression, context window expansion, and CoT overthinking.

4.1 KV Cache Compression

The quadratic growth of the Key-Value (KV) cache is a primary bottleneck in LLM inference. To bound memory, heuristic eviction methods like StreamingLLM (Xiao et al., 2023) and H2O (Zhang et al., 2023) preserve "anchor" or high-attention tokens. SnapKV (Li et al., 2024) and PyramidKV (Cai et al., 2025) refine this via saliency clustering and layer-wise adaptive budgets. Other strategies include reconstruction-based methods like KVzip (Kim et al., 2025) and low-bit quantization such as KIVI (Liu et al., 2024). However, lossy compression can induce "distraction" (Chen et al., 2025a), shifting attention to irrelevant content and degrading performance on precise reasoning tasks. Unlike these throughput-centric approaches, Free()LM implements logic-aware pruning. By training a Free-Module to detect semantic redundancy within reasoning traces, Free()LM improves efficiency and maintains reasoning quality.

4.2 Long Context Window

Expanding the effective context window is a major research frontier. Early architectural innovations modified positional encodings to extend context limits. ALiBi (Press et al., 2022) replaces absolute positions with relative biases to improve length extrapolation. YaRN (Peng et al., 2024) and LongRoPE (Ding et al., 2024) extend RoPE via rescaling/interpolation, enabling million-token contexts and pushing theoretical limits beyond 2M tokens. System-level approaches such as Ring Attention (Liu et al., 2023) distribute attention computation across GPUs using blockwise schemes, further increasing feasible context length. Training-based methods such as LongLoRA (Chen et al.,

Free()LM: Stable Deletion

[Current step]
According to previous analysis, two non-empty intersections. However, the problem states three regions. Therefore, perhaps I made an error in considering Case 2 ∩ Case B?
Wait, let me check again.
...
Therefore, Case 2 ∩ Case B is empty.
Therefore, the only non-empty intersections are Case 1 ∩ Case B and Case 2 ∩ Case A. But the problem states three regions. So, where is the third region?
[Next step](continues after deletion)
Suppose we consider the entire plane $x + y + z = 75$. The inequalities divide it into regions. Since the problem says three regions, maybe there's another region where both inequalities hold? ...

Gemini Deletion: Regeneration

[Current step]
Alternatively, since Γ is the nine-point circle, maybe there are some symmetries?
Maybe use coordinates
...
find circumcircle of D, E, F
Let me find the equation of the circle passing through these three points.
[Next step](re-generated)
Also, angle at B is 60°, which can be used to find coordinates.
Maybe use coordinates
...
find circumcircle of D, E, F
Let me find the equation of the circle ...

Figure 5: Case study comparing Free()LM versus Gemini deletion. Deleted spans are shown in red, new generated content in green, and re-generated content matching previous deletions in blue. Free()LM (left) successfully removes redundant reasoning with stable deletion; Gemini (right) regenerates previously deleted content, undermining the deletion effectiveness.

2024) adapt pre-trained models to longer contexts efficiently. Despite these advances, Du et al. (2025) show performance can degrade as context grows due to distraction, even with perfect retrieval; Modarressi et al. (2025) similarly reports accuracy decay beyond 32k tokens across diverse reasoning tasks. This is orthogonal but complementary to Free()LM: rather than expanding the window, we address the "memory leak" by teaching models to actively free() obsolete information.

4.3 Overthinking

While Chain-of-Thought (CoT) (Wei et al., 2022) improves LLM reasoning, it can induce overthinking: excessive, redundant steps that consume context without improving the final answer (Chiang and Lee, 2024). This is especially acute in multi-step math, where models may explore dead ends, repeat computations, or over-verify. The rise of o1-style long-reasoning models intensifies this issue, with 10k–100k+ tokens for a single problem; Chen et al. (2025b) shows redundancy can appear even in trivial arithmetic. Prior work mitigates overthinking via budgeted generation (Han et al., 2025), pruning based on token importance (Choi et al., 2025), or stopping rules such as entropy monitoring (Jiang et al., 2025). Other methods optimize conciseness through training, e.g., O1-Pruner (Luo et al., 2025) uses reinforcement learning to reward shorter correct reasoning, and DeepCompress (Liang et al., 2025) employs adaptive length rewards based on problem difficulty. Unlike methods that primarily control output length, Free()LM enables dynamic context cleaning during inference through a trainable Free-Module that free()s redundant reasoning steps, keeping the workspace efficient throughout generation.

5 Conclusion

We argue that the prevailing "malloc-only" paradigm is fundamentally unsustainable for infinite reasoning. By treating the context window as a passive, append-only buffer, standard LLMs allow noise to accumulate until it overwhelms the model's attention, leading to the inevitable "cognitive memory leak." In this work, we bridge this gap by introducing **Free()LM**, which completes the memory management cycle with the missing free() operation.

By transforming the context from a static history into a dynamic, managed workspace, Free()LM ensures sustained reasoning stability—a critical capability beyond simple context scaling. Our results demonstrate that this active management not only improves performance across the board (averaging +4.4%) but also prevents catastrophic failure in extreme regimes. On the challenging HLE benchmark in Qwen3-235B-A22B, specifically for cases requiring 70k+ tokens, Free()LM boosts accuracy from a flat 0% to 28% while halving memory consumption.

Ultimately, Free()LM redefines the scaling laws of test-time compute. We demonstrate that the path to infinite reasoning lies not merely in expanding the context window, but in mastering the art of forgetting. This shift from "malloc-only" to "malloc + free" lays the foundation for the next generation of efficient, self-sustaining agentic systems.

Limitations

Free()LM currently triggers Clean-Up at a fixed interval, which may be suboptimal across problem types: some instances need more frequent deletion to avoid context pollution, while others benefit from leaving the context untouched. Our main serving implementation also relies on re-prefilling after deletion for compatibility with standard inference stacks; although KV-cache pruning could reduce this overhead, it is not yet integrated in our setup. Finally, training uses rollout-based verification to ensure deletions preserve accuracy, which is computationally expensive and may bias learning toward domains with reliable automatic checks. Despite these limitations, Free()LM remains lightweight and plug-and-play, delivering consistent gains and substantial memory savings while keeping the backbone frozen—providing a practical foundation for future improvements.

Ethics Statement

Free()LM studies context management for long-horizon reasoning using publicly accessible datasets and model-generated traces. Our research does not involve human subjects studies, personally identifiable information, or any interaction with private user data. We do not anticipate additional ethical concerns arising from this work. Lastly, AI was used to revise the grammar during the paper writing process.

References

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. 2025a. Matharena: Evaluating llms on uncontaminated math competitions.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. 2025b. Matharena: Evaluating llms on uncontaminated math competitions.

ByteDance-Seed. 2025. Beyondaime: Advancing math reasoning evaluation beyond high school olympiads. https://huggingface.co/datasets/ByteDance-Seed/BeyondAIME.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. 2025. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. In *Second Conference on Language Modeling*.

Alex Chen, Renato Geh, Aditya Grover, Guy Van den Broeck, and Daniel Israel. 2025a. The pitfalls of kv cache compression. *Preprint*, arXiv:2510.00231.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025b. Do NOT think that much for 2+3=? on the overthinking of long reasoning models. In *Forty-second International Conference on Machine Learning*.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2024. Longlora: Efficient fine-tuning of long-context large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Cheng-Han Chiang and Hung-yi Lee. 2024. Over-reasoning and redundant calculation of large language models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 161–169, St. Julian's, Malta. Association for Computational Linguistics.

Daewon Choi, Jimin Lee, Jihoon Tack, Woomin Song, Saket Dingliwal, Sai Muralidhar Jayanthi, Bhavana Ganesh, Jinwoo Shin, Aram Galstyan, and Sravan Babu Bodapati. 2025. Think clearly: Improving reasoning via redundant token pruning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 21437–21451, Suzhou, China. Association for Computational Linguistics.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, and Evan Rosen et. al. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *Preprint*, arXiv:2507.06261.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, and Chengda Lu et. al. 2025. Deepseek-v3.2: Pushing the frontier of open large language models. *Preprint*, arXiv:2512.02556.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. 2024. Longrope: Extending llm context window beyond 2 million tokens. *Preprint*, arXiv:2402.13753.

Yufeng Du, Minyang Tian, Srikanth Ronanki, Subendhu Rongali, Sravan Babu Bodapati, Aram Galstyan, Azton Wells, Roy Schwartz, Eliu A Huerta, and Hao Peng. 2025. Context length alone hurts LLM performance despite perfect retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 23281–23298, Suzhou, China. Association for Computational Linguistics.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. Token-budget-aware LLM reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24842–24855, Vienna, Austria. Association for Computational Linguistics.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *Preprint*, arXiv:2504.11456.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Tianyi Jiang, Yi Bin, Yujuan Ding, Kainian Zhu, Fei Ma, Jingkuan Song, and Heng Tao Shen. 2025. Explore briefly, then decide: Mitigating llm overthinking via cumulative entropy regulation. *Preprint*, arXiv:2510.02249.

Andrej Karpathy. 2023. Llm as an operating system. `https://twitter.com/karpathy/status/1723140519554105733`. Accessed: 2025-12-04.

Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W Lee, Sangdoo Yun, and Hyun Oh Song. 2025. Kvzip: Query-agnostic kv cache compression with context reconstruction. *arXiv preprint arXiv:2505.23416*.

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

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.

Tian Liang, Wenxiang Jiao, Zhiwei He, Jiahao Xu, Haitao Mi, and Dong Yu. 2025. Deepcompress: A dual reward strategy for dynamically exploring and compressing reasoning chains. *Preprint*, arXiv:2510.27419.

Hao Liu, Matei Zaharia, and Pieter Abbeel. 2023. Ring attention with blockwise transformers for near-infinite context. *arXiv preprint arXiv:2310.01889*.

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

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. 2025. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *Preprint*, arXiv:2501.12570.

Thang Luong, Dawsen Hwang, Hoang H. Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, Alex Zhai, Clara Huiyi Hu, Henryk Michalewski, Jimin Kim, Jeonghyun Ahn, Junhwi Bae, Xingyou Song, Trieu H. Trinh, Quoc V. Le, and Junehyuk Jung. 2025. Towards robust mathematical reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*.

Dongyang Ma, Yan Wang, and Tian Lan. Block-attention for efficient prefilling. In *The Thirteenth International Conference on Learning Representations*.

Ali Modarressi, Hanieh Deilamsalehy, Franck Dernoncourt, Trung Bui, Ryan A. Rossi, Seunghyun Yoon, and Hinrich Schütze. 2025. Nolima: Long-context evaluation beyond literal matching. *Preprint*, arXiv:2502.05167.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2024. YaRN: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, and Hugh Zhang et. al. 2025. Humanity's last exam. *Preprint*, arXiv:2501.14249.

Ofir Press, Noah A. Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, , and Jason Wei. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024. Mmlu-pro: a more robust and challenging multi-task language understanding benchmark. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA. Curran Associates Inc.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

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, and Chujie Zheng et. al. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Yifan Zhang and Team Math-AI. 2024. American invitational mathematics examination (aime) 2024.

Yifan Zhang and Team Math-AI. 2025. American invitational mathematics examination (aime) 2025.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, and 1 others. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710.

A Experimental Details

A.1 Prompts

The prompts utilized in this study are detailed in Figure 6. We employ a consistent prompt for both training corpus preparation and alignment evaluation. To ensure a structured response format, we implement a JSON schema to guide the Free-Module's output, as illustrated in Figure 7. Both vLLM (Kwon et al., 2023) and Gemini natively support this structured output functionality.

A.2 Training Details

Since `Free()LM` is trained to perform deletion over intermediate reasoning produced by backbone models that do not expose explicit reasoning traces as supervision, we disable the *thinking mode* in the chat template during prompt construction. This design ensures that the Free-Module operates solely on the visible model-generated reasoning, matching the inference-time setting.

We train the Free-Module on the Qwen3-8B backbone using TRL (v0.19.1) (von Werra et al., 2020), while the larger Qwen3-30B-A3B and Qwen3-235B-A22B variants are trained with Megatron-LM (v0.14.1) (Shoeybi et al., 2019). All models are trained for 5 epochs with a learning rate of 1×10^{-5} and a global batch size of 16. To optimize memory efficiency and throughput, we use the AdamW optimizer together with FlashAttention-2 for attention computation. We employ DeepSpeed ZeRO Stage 3 for Qwen3-8B and Stage 2 for Qwen3-30B-A3B and Qwen3-235B-A22B, balancing memory savings and communication overhead across model scales. All training runs are conducted using BF16 precision.

For parameter-efficient fine-tuning, we apply LoRA to all linear layers with rank $r = 128$, scaling factor $\alpha = 256$, and dropout rate 0.1. The backbone model parameters remain frozen throughout training; only the Free-Module parameters are updated.

A.3 Evaluation Details

For prefilling, we append the model's previously generated response to the end of the prompt after applying the chat template, and re-run the prompt through the model to resume generation from the refined context. This procedure ensures that subsequent tokens are generated conditionally on the updated reasoning trace, rather than restarting inference from scratch.

In practice, we leverage vLLM's support for efficient prefilling and KV cache reuse. When the context is modified by a `free()` operation, we reuse the cached key–value states corresponding to the unchanged prefix, and only re-prefill the altered suffix. This avoids recomputing attention for the entire context and significantly reduces the overhead of resuming inference.

System Prompt:
Role: AI Reasoning Analyst
Your task is to act as an AI Reasoning Analyst. You will be given a Chain-of-Thought (CoT) reasoning and your goal is to identify and mark for deletion any paragraphs from the CoT reasoning that are redundant or irrelevant.
Instructions:

1. **Analyze the CoT Reasoning:** Carefully read and understand the reasoning, context, and goals of the CoT reasoning.
2. **Identify Redundant or Irrelevant Paragraphs:** A paragraph is considered redundant or irrelevant if it does not directly support or inform the overall reasoning process. This could include tangential thoughts, corrected errors, or superseded lines of reasoning.
3. **Mark for Deletion:** For every irrelevant paragraph you detect, generate a JSON object that uniquely identifies it. Each object must include a `prefix` and a `suffix` extracted from the paragraph.
4. **Format the Output:** Your final output must be a single JSON object containing a list of these `prefix`/`suffix` objects.
5. `<DELETED>` in the CoT reasoning indicates that the content has already been removed.
6. It is possible that you don't need to delete any paragraphs in the CoT reasoning.

Constraints:

- If no paragraphs are redundant or irrelevant, output an empty list: `[]`.
- If multiple paragraphs are redundant, include a separate JSON object for each one.
- Do not delete the first or the last paragraph in the CoT reasoning.

User Prompt:

```
### CoT Reasoning:
<PREVIOUS_COT>
```

Figure 6: **Deletion Prompts for Free()LM.** The system prompt (top) provides the operational logic for identifying redundant reasoning steps, while the user prompt (bottom) delivers the actual CoT context for the agent to process.

Response Schema (JSON):

```
{
   "type": "array",
   "items": {
      "type": "object",
      "properties": {
         "prefix": {
            "type": "string",
            "description": "Beginning text of paragraph to delete."
         },
         "suffix": {
            "type": "string",
            "description": "Ending text of paragraph to delete."
         }
      },
      "required": ["prefix", "suffix"]
   },
   "description": "A list of prefix/suffix pairs that uniquely identify paragraphs that are redundant
or irrelevant and should be deleted."
}
```

Figure 7: **Structured Output Schema for Free()LM.** The JSON schema enforces a consistent prefix/suffix format, ensuring that redundant or irrelevant reasoning steps identified by the agent can be programmatically parsed and removed from the context.