Dynamic Ledger: Retrieval-Augmented Structured Memory for Test-Time Learning

Stanford CS224N Custom Project

Webpage: `https://web.stanford.edu/~srliu/ai/`

Jerry Gu
Department of Computer Science
Stanford University
jerrygu@stanford.edu

Shurui Liu
Department of Mathematics
Stanford University
srliu@stanford.edu

Sabrina Yen-Ko
Department of Computer Science
Stanford University
syenko@stanford.edu

We take **2 late days (6 together)** with the following distribution: 3 from Shurui Liu, 3 from Sabrina Yen-Ko, 0 from Jerry Gu

Abstract

While modern language models excel at many complex tasks, they lack a persistent, adaptable memory to learn new problem-solving strategies during test-time. We introduce two extensions to the original Dynamic Cheatsheet framework [1]: **Strategic Chunk Retrieval**(SCR) and **Dynamic Ledger**(DL). **SCR** replaces the monolithic cheatsheet with a structured, chunk-level memory store where each entry is an independently retrievable strategy unit, combating strategy dilution and irrelevant-context contamination that arise when the cheatsheet is scaled. **DL** converts this memory store into a structured database with CRUD operations and dual-retrieval on both the strategy and originating problem embedding. We evaluate both extensions on GPT-4o (and GPT5) across 4 benchmarks—AIME 2020–2024, MathEquationBalancer, IneqMath, and DataSIR—and find that both DL and SCR with top-k=3 selection consistently improves over all baselines from [1], with the largest gains on math-intensive tasks where strategy reuse is most beneficial. Finally, a sensitivity analysis confirms that reasoning performance degrades significantly under strategy dilution, empirically validating the necessity of our selective curation approach.

1 Key Information

- **TA Mentor:** Mirac Suzgun
- **External Collaborators:** No
- **External Mentor:** No
- **Sharing Project:** No

2 Introduction

Large language models (LLMs) have demonstrated impressive reasoning capabilities across mathematics, science, and programming benchmarks, yet they remain fundamentally stateless at inference time: every new problem is solved from scratch, discarding hard-won insights from prior queries. This contrasts sharply with human problem-solving, where accumulated experience and learned heuristics dramatically accelerate performance on unfamiliar but structurally related problems.

The Dynamic Cheatsheet (DC) framework [1] addresses this gap by endowing black-box LLMs with a persistent, self-curated memory that grows during inference. A *generator* solves each current problem using the cheatsheet, and a *curator* updates it from the experience. Two main variants exist: **DC-Cu** (cumulative), which appends all experience sequentially, and **DC-RS** (retrieval synthesis), which retrieves similar past problems and synthesizes a query-specific cheatsheet before generation. Both substantially outperform stateless baselines on mathematical and scientific reasoning tasks.

Despite these gains, DC has two notable limitations. First, the cheatsheet is maintained as a *monolithic text artifact*: the curator must reconcile all stored strategies in each single pass, which can dilute task-specific strategies and bury relevant knowledge under irrelevant content as the cheatsheet grows–an instance of the "lost in the middle" effect [2]. Second, retrieval in DC-RS is *quality-agnostic*: incorrect past solutions are treated identically to correct ones during retrieval, allowing confidently wrong examples to mislead the generator.

We propose two targeted extensions that directly address these limitations. **Strategic Chunk Retrieval** (Section 4.3) replaces the monolithic cheatsheet with a structured, chunk-level memory store of self-contained strategy units, retrieved by content similarity to the query and updated selectively—leaving unrelated strategies untouched. **Dynamic Ledger** (Section 4.4) reframes this store as a lightweight database supporting explicit CRUD operations and dual-embedding retrieval over both strategy text and source-problem embeddings, recovering relevant entries that strategy-only retrieval misses. We additionally explore **Confidence-Weighted Retrieval** (Section 4.2), which re-ranks retrieved examples by an ensemble-based trust score, filtering low-confidence.

We evaluate all methods on four benchmarks—AIME 2020–2024, IneqMath, MathEquationBalancer, and DataSIR—and find that Dynamic Ledger consistently achieves the highest accuracy, with the largest gains on math-intensive tasks where strategy reuse is most beneficial (Section 5.1.4). A sensitivity analysis further confirms that performance degrades significantly under strategy dilution, empirically validating the need for selective curation (Section 5.2). Confidence-Weighted Retrieval did not independently improve over DC baselines; we discuss this result and directions for integration in Section 7.

3 Related Work

Dynamic Cheatsheet. Our work directly builds on [1], which introduced the DC framework and demonstrated large gains on AIME, GPQA, Game of 24, and other benchmarks. A concurrent extension, Agentic Context Engineering [3], explores self-improving context management over longer agent episodes; our approach differs by targeting retrieval quality and memory granularity rather than context scheduling.

Memory management for LLM agents. MemGPT [4] introduced a hierarchical memory model (in-context + external archival) with LLM-driven paging, motivating the idea of selective memory access. A-MEM [5] adopts a Zettelkasten-style atomic-note architecture with two-stage retrieval, closely related to our chunk-level memory store. ReasoningBank [6] distills strategies from agent trajectories into a structured memory bank, the most directly comparable prior work to Strategic Chunk Retrieval. Plan Caching [7] caches reusable plan templates from successful runs for keyword-based lookup, a complementary strategy at a coarser plan granularity. A broad survey of agent memory and tool use efficiency is provided by [8].

Self-assessment and confidence estimation. Kadavath et al. [9] showed that LLM self-reported confidence is overconfident and imperfect, motivating our move from self-assessment to ensemble-based trust scoring. Reflexion [10] uses verbal reinforcement—reflecting on past failures to avoid repeating them—which shares the spirit of our trust-based filtering but operates at the trajectory level

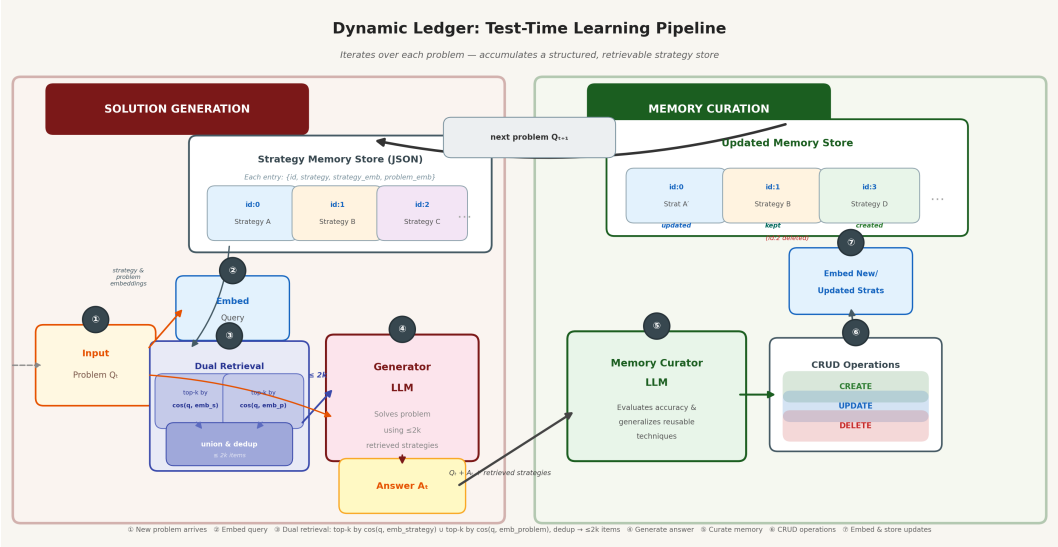


Figure 1: Overview of the Dynamic Ledger test-time learning pipeline

rather than the retrieval-ranking level. Generative Agents [11] pioneered multi-factor memory scoring combining importance, recency, and relevance, directly inspiring our multi-factor re-ranking formula.

Retrieval-augmented generation. RAG [12] established the paradigm of augmenting generation with retrieved external knowledge; our work applies a similar retrieval principle but over a *self-generated, evolving* strategy store rather than a static corpus. The finding that LLMs attend poorly to information in the middle of long contexts [2] further motivates our selective curation design, which avoids flooding the prompt with irrelevant strategies.

4 Approach

4.1 Baseline: Dynamic Cheatsheet with Retrieval

Our work builds on the DC framework [1]. We extend **DC-RS** with Confidence-Weighted Retrieval and **DC-Cu** with Strategic Chunk Retrieval, and evaluate against the baselines **DC-** (no cheatsheet), **DC-Cu**, and **DC-RS** from the original paper on GPT-4o [13]. We additionally evaluate on **IneqMath** [14], an inequality-proof benchmark not present in the original DC evaluation.

Formally, let $\mathbf{e}_i \in \mathbb{R}^d$ denote the embedding of input x_i. For the n-th question, DC-RS retrieves:

$$\mathcal{R}_n^{\text{top-}k} := \text{argmax}^{(k)} \{ i < n : \cos\langle \mathbf{e}_n, \mathbf{e}_i \rangle \}$$

and constructs the cheatsheet from the retrieved input–output pairs $\{(x_i, y_i) : i \in \mathcal{R}_n^{\text{top-}k}\}$ via a separate curator LLM call.

We also implement a *probability-cut* variant that converts cosine similarities to a softmax distribution $p_{i,n}$ and retrieves the minimal set covering cumulative probability p:

$$\mathcal{R}_n^{\text{prob-}p} := \text{argmin}_{R \subseteq \{1,\dots,n-1\}} \{ |R| : \textstyle\sum_{j \in R} p_{j,n} \geq p \}.$$

4.2 Confidence-Weighted Retrieval

A core limitation of similarity-only retrieval is that incorrect solutions are treated identically to correct ones: a confidently wrong example can mislead the generator. We explore **confidence-weighted re-ranking** scheme that augments retrieval with a trust score derived *without* ground-truth labels.

Ensemble-based trust scoring. We sample N additional independent responses $(\tilde{y}_i^{(j,\tau_j)})_{j=1}^N$ at varying temperatures $\tau_j \in \{0.7, 0.8, 0.9\}$ after generating $\hat{y}_i$. The trust score is the agreement ratio:

$$s_i^{\text{trust}} = \frac{1}{N} \sum_{j=1}^{N} \mathbf{1}\left[\tilde{y}_i^{(j,\tau_j)} = \hat{y}_i\right],$$

where answers are extracted via the existing structured parser. High agreement across temperatures indicates a stable, reliable response.

Multi-factor re-ranking. We replace similarity-only ranking with a weighted combination:

$$\text{score}(i, n) = w_{\text{sim}} \cdot \cos\langle \mathbf{e}_n, \mathbf{e}_i \rangle + w_{\text{trust}} \cdot s_i^{\text{trust}},$$

with default weights $(w_{\text{sim}}, w_{\text{trust}}) = (0.5, 0.5)$. Any example with $s_i^{\text{trust}} < 0.5$ is excluded from the candidate pool before selection, preventing low-quality examples from entering the cheatsheet synthesis stage entirely.

4.3 Strategic Chunk Retrieval

We propose **Strategic Chunk Retrieval**, a memory architecture that replaces the monolithic cheatsheet with a structured, chunk-level memory store and retrieves items based on their *full strategy content* rather than only their source question.

Chunk-level memory store. Each entry in the store is a single `<memory_item>`: a self-contained strategy, code snippet, or insight. Every item carries metadata $m_i = (\text{text}_i, \mathbf{v}_i, c_i)$, where $\mathbf{v}_i \in \mathbb{R}^d$ is the embedding of the strategy text itself and $c_i \in \mathbb{N}$ is a usage counter initialized to 1.

Content-based retrieval with usage bonus. Given the n-th input x_n with embedding $\mathbf{e}_n$, we score every memory item:

$$\text{score}(i, n) = \alpha \cdot \cos\langle \mathbf{e}_n, \mathbf{v}_i \rangle + (1 - \alpha) \cdot \frac{\ln(1 + c_i)}{\ln(1 + c_{\max})}, \tag{1}$$

with $\alpha = 0.85$ and $c_{\max} = \max_j c_j$. The logarithmic normalization prevents high-count items from dominating while still surfacing battle-tested strategies over untested alternatives. We support **Top-k** selection and a **Probability threshold (p)** variant analogous to Section 4.1. After retrieval, each selected item's usage counter is incremented: $c_i \leftarrow c_i + 1$.

Selective curation. The curator receives *only the retrieved chunks*, not the full store, and produces an updated set of `<memory_item>` blocks. Retrieved items are updated and re-embedded on their strategy text; non-retrieved items remain untouched. This localization prevents information loss in the broader store and keeps the curator prompt short and focused.

4.4 Dynamic Ledger

Strategic Chunk Retrieval (Section 4.3) has two remaining limitations. First, its document-centric curation model rewrites retrieved items as a batch of text blocks so an update to one strategy may inadvertently reword another. Second, retrieval only matches on strategy content, missing cases where a different-looking technique applies to a structurally similar problem.

Dynamic Ledger addresses both by reframing the cheatsheet as a lightweight database with explicit CRUD operations and dual-embedding retrieval.

Structured memory entries. Each entry in the ledger is a JSON record $r_i = \big(\texttt{id}_i,\ \text{strategy}_i,\ \text{example}_i,\ \mathbf{s}_i,\ \mathbf{p}_i\big)$, where $\texttt{id}_i \in \mathbb{N}$ is the unique identifier, strategy_i is the core problem-solving technique, example_i records the originating problem, $\mathbf{s}_i \in \mathbb{R}^d$ is the embedding of the strategy text, and $\mathbf{p}_i \in \mathbb{R}^d$ is the embedding of the example problem. Separating and independently embedding strategy and problem enables dual-axis retrieval below. We also remove the usage counter: qualitative analysis of preliminary results showed that it biases the store toward heavily generalized but vague strategies.

Dual-embedding retrieval. Given a new input x_n with embedding $\mathbf{e}_n$, we retrieve along both axes independently:

$$\mathcal{R}_{\text{prob}} = \text{Top-k}\big(\cos\langle \mathbf{e}_n,\, \mathbf{p}_i \rangle\big), \tag{2}$$

$$\mathcal{R}_{\text{strat}} = \text{Top-k}\big(\cos\langle \mathbf{e}_n,\, \mathbf{s}_i \rangle\big), \tag{3}$$

and return the deduplicated union $\mathcal{R}_n = \mathcal{R}_{\text{prob}} \cup \mathcal{R}_{\text{strat}}$, yielding at most $2k$ items. Problem-axis retrieval (2) surfaces strategies that were born from structurally similar questions, even when the strategy text itself is phrased abstractly. Strategy-axis retrieval (3) captures techniques whose *description* matches the current need, regardless of the originating problem's surface form. The union of both sets provides broader recall than either axis alone.

CRUD-based curation. Rather than regenerating retrieved items as a revised text block, the curator emits a structured JSON array of atomic operations:

- CREATE: append a new entry with a fresh `id`; the system embeds both the strategy text and the current problem to produce $\mathbf{s}_i$ and $\mathbf{p}_i$.
- UPDATE: replace the strategy text of an existing entry by `id`; $\mathbf{s}_i$ is re-embedded while $\mathbf{p}_i$ is preserved.
- DELETE: remove an entry by `id`, used only when a strategy is demonstrably incorrect or fully subsumed.

This explicit operation language gives the curator fine-grained, auditable control over each record. Non-retrieved entries are never exposed to the curator and cannot be accidentally altered, extending the locality guarantee of Section 4.3 while additionally preventing unintended cross-entry interference within the retrieved set.

5 Experiments

5.1 Benchmark Evaluation

5.1.1 Data

We evaluate on four benchmarks spanning different reasoning modalities:

- **AIME 2020–2024** (133 problems): American Invitational Mathematics Examination competition problems from 2020–2024, each requiring a single integer answer.
- **IneqMath** (100 problems) [14]: An inequality-proof benchmark requiring formal mathematical reasoning. Not evaluated in the original DC paper, we include it to stress-test strategy transfer under a qualitatively different task structure.
- **MathEquationBalancer** (250 problems): Procedural chemical equation balancing tasks with exact numerical coefficient answers.
- **DataSIR** (100 problems) [15]: A sensitive information recognition benchmark that tests pattern-matching and structured data analysis.

5.1.2 Evaluation Method

We follow the evaluation protocol of Suzgun et al. [1]: problems are presented sequentially, the cheatsheet evolves across the sequence, and accuracy is measured as the fraction of correctly answered problems. For AIME and MathEquationBalancer, correctness is determined by exact string match after extracting the final numerical answer. For IneqMath, we use type-specific evaluation: letter extraction for relation problems and normalized expression matching for bound problems. For DataSIR, we apply exact match on the extracted answer. All methods see problems in the same randomly shuffled order (seed 10).

We compare against four baselines from [1]: **Baseline** (no cheatsheet), **EmptyCheatsheet** (empty cheatsheet passed to generator), **DC-Cu** (cumulative cheatsheet), and **DC-RS** (retrieval synthesis). We additionally report **FullHistoryAppend** (verbatim Q&A history) and **Dynamic Retrieval** (top-k past pairs without synthesis) as reference points.

Table 1: Accuracy (%) across four benchmarks. The first three benchmarks use GPT-4o; DataSIR uses GPT-5. Best result per column is **bolded**. "—" indicates the method was not evaluated on that benchmark. Methods above the mid-rule are baselines; methods below are ours.

Method	**AIME** (133, 4o)	**IneqMath** (100, 4o)	**MathEqBal** (250, 4o)	**DataSIR** (100, 5)
Baseline / Default	9.8	48.0	47.2	87.0
EmptyCheatsheet	24.1	—	83.2	—
DC-Cu	18.0	47.0	94.4	84.0
FullHistoryAppend	—	—	—	88.0
Dynamic Retrieval	24.1	—	94.4	—
DC-RS	24.8	47.0	94.0	87.0
DC-SCR (ours)	28.2	53.0	**100.0**	75.0
DC-SCR p=0.8 (ours)	20.6	49.0	—	—
DC-DL (ours)	**30.8**	**58.0**	**100.0**	**91.0**

5.1.3 Experimental Details

All primary experiments use **GPT-4o** [13] as both the generator and curator, with temperature 0.0 and a maximum of $4{,}096$ output tokens. DataSIR is evaluated on GPT-5 for cost efficiency on the 100-problem subset. The generator executes a single reasoning round per problem (`max_num_rounds` $= 1$), with Python code execution enabled for problems that benefit from computation.

For all retrieval-based methods, question embeddings are pre-computed using OpenAI's `text-embedding-3-small` (d=1,536). Strategic Chunk Retrieval and Dynamic Ledger additionally embed strategy text at curation time using the same model. We use top-k=3 retrieval for all retrieval-based methods unless otherwise noted. The blending coefficient is α=0.85 for Strategic Chunk Retrieval (Eq. 1). We also evaluate a probability-threshold variant (p=0.8) for Strategic Chunk Retrieval on AIME and IneqMath to study adaptive retrieval cardinality.

Our implementation extends the open-source Dynamic Cheatsheet codebase[1] with new modules for chunk-level memory management, CRUD-based ledger operations, dual-embedding retrieval, and ensemble trust scoring. The curator prompts for Strategic Chunk Retrieval and Dynamic Ledger are provided in Appendix A.

5.1.4 Results

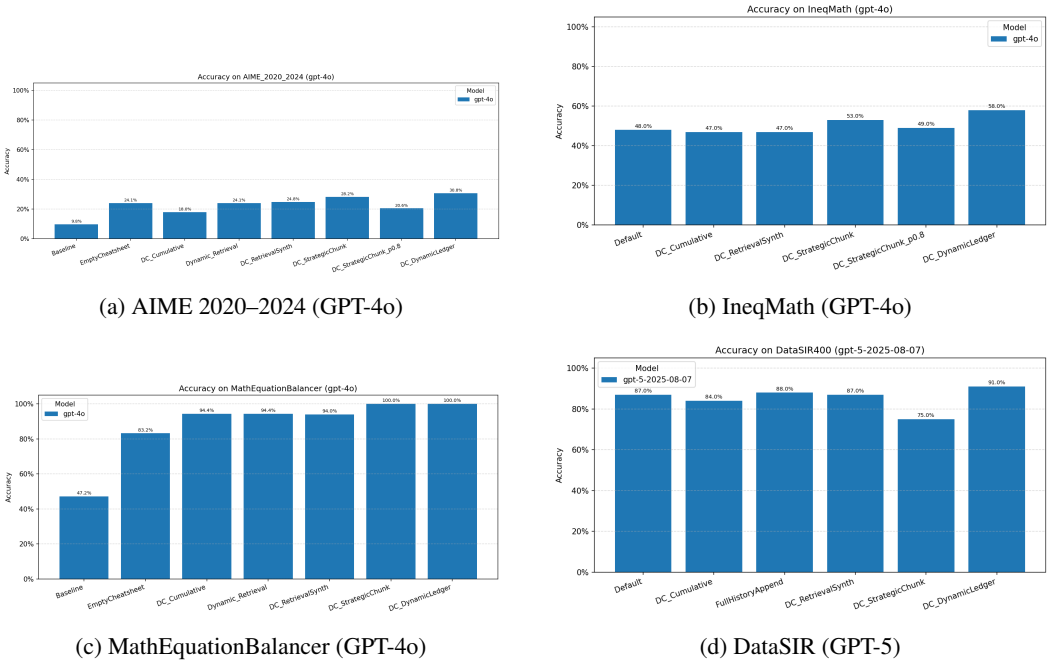
Table 1 and Figure 2 present accuracy across all four benchmarks.

AIME 2020–2024. DC-DL achieves the highest accuracy at 30.8%, a 3.1× improvement over the stateless Baseline (9.8%) and a 6.0 percentage point gain over the best prior DC variant, DC-RS (24.8%). DC-SCR (28.2%) also surpasses all baselines, confirming that chunk-level memory management yields more effective strategy reuse on competition math. The probability-threshold variant DC-SCR p=0.8 (20.6%) underperforms the top-k variant, suggesting that adaptive cardinality introduces too many marginally relevant chunks that dilute the generator's prompt.

IneqMath. DC-DL leads at 58.0%, a 10 percentage point gain over the Default baseline (48.0%), while DC-SCR reaches 53.0%. Notably, DC-Cu and DC-RS both underperform the Default (47.0% each), indicating that monolithic curation and similarity-only retrieval offer no benefit—and may even hurt—on inequality proof tasks. The strong performance of chunk-based methods suggests that fine-grained strategy retrieval is particularly advantageous when problems require diverse, specialized proof techniques.

MathEquationBalancer. Both DC-SCR and DC-DL achieve perfect accuracy (100.0%), a dramatic improvement over the 47.2% Baseline. The large jump from Baseline to EmptyCheatsheet (83.2%) indicates that even a minimal structured prompt substantially helps. Prior DC variants plateau around 94%, unable to close the remaining gap. The perfect score from chunk-based methods shows that

[1] `https://github.com/suzgunmirac/dynamic-cheatsheet`



(a) AIME 2020–2024 (GPT-4o)

(b) IneqMath (GPT-4o)

(c) MathEquationBalancer (GPT-4o)

(d) DataSIR (GPT-5)

Figure 2: Accuracy comparison across four benchmarks. DC-DL (Dynamic Ledger) achieves the highest accuracy on all four tasks. DC-SCR (Strategic Chunk Retrieval) provides strong gains on GPT-4o math benchmarks but underperforms on DataSIR due to single-axis retrieval limitations.

for tasks with highly reusable procedural strategies, structured memory eliminates residual errors entirely.

DataSIR. On this benchmark (evaluated with GPT-5 medium thinking), DC-DL achieves 91.0%, outperforming all methods. However, DC-SCR drops to 75.0%, well below the Default (87.0%). This reversal highlights the importance of dual-embedding retrieval: when strategy text alone does not capture the structural similarity between problems, the problem-embedding channel in DC-DL compensates, recovering relevant past entries that the strategy-only channel misses.

5.2 Sensitivity Analysis: Strategy Dilution

To isolate the impact of **contextual interference** independently of retrieval quality, we conduct a controlled sensitivity analysis on the AIME 2021–2025 benchmark [16]. We follow the same evaluation method described in 5.1.2.

5.2.1 Experiment Details

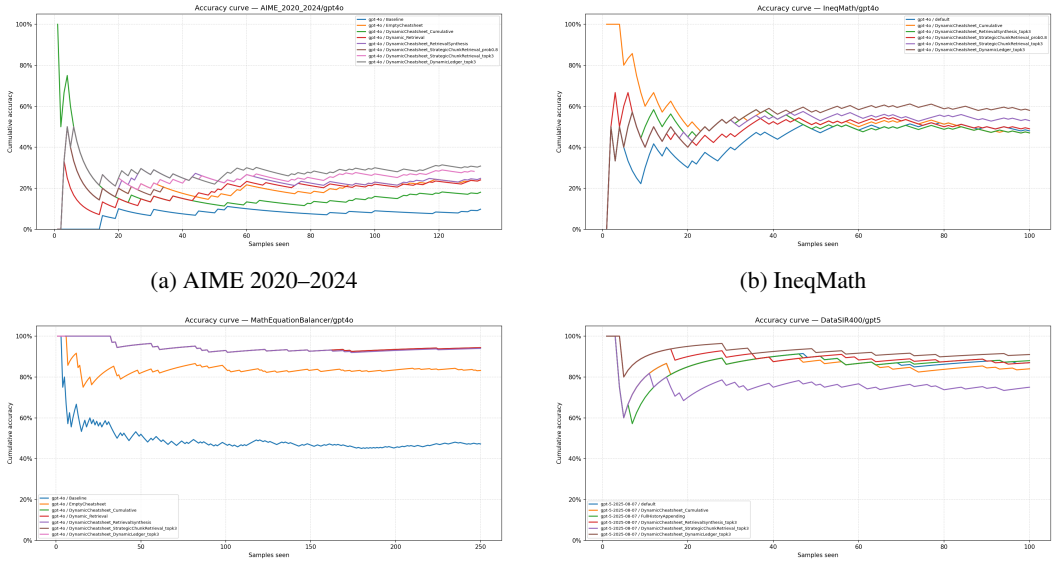
We utilize an **oracle retrieval** protocol where the generator is provided with exactly one "gold" memory item: the correct strategy for the specific problem. This memory item is generated directly based on the provided solution from the dataset. We then vary the **distractor cardinality** $n \in \{0, 10, 50\}$ by injecting strategies from unrelated problems from the same benchmark. To mitigate positional bias, items are randomly shuffled before being passed to the generator.

5.2.2 Results

Accuracy exhibits a consistent decay as the number of distractors increases (Table 2). From an oracle baseline of 22.5%, performance drops significantly, reaching 15.8% under high-noise conditions ($n = 50$).

Table 2: Sensitivity of GPT-4o on AIME 2021–2025. Accuracy is measured using an oracle retrieval setup ($n = 0$ is the gold strategy).

Configuration	**Distractors** (n)	**Accuracy (%)**	**Relative Decay**
Oracle Strategy	0	22.5%	—
Low Distraction	10	19.2%	−14.7%
High Distraction	50	15.8%	−29.8%



(a) AIME 2020–2024

(b) IneqMath

(c) MathEquationBalancer

(d) DataSIR

Figure 3: Cumulative accuracy curves over the problem sequence. All DC variants start from the same accuracy and diverge as the memory store accumulates strategies. DC-DL (pink/brown) consistently maintains the highest cumulative accuracy after an initial learning phase.

6 Analysis

6.1 Cumulative Learning Curves

Figure 3 plots cumulative accuracy as a function of problems seen, revealing the *test-time learning dynamics* of each method. On AIME (Figure 3a), the Baseline flatlines near 10% while DC-DL and DC-SCR separate from DC baselines around problem 40 and maintain a widening gap; DC-Cu's early convergence below DC-RS reflects the monolithic curation bottleneck. MathEquationBalancer (Figure 3c) provides the clearest illustration: DC-SCR and DC-DL reach near-perfect accuracy within the first 20 problems, while DC-Cu and DC-RS plateau near $\sim$94% and the memoryless Baseline stalls at $\sim$47%. On DataSIR (Figure 3d), DC-SCR remains below all methods throughout, while DC-DL climbs to the top by problem 30, confirming that dual-embedding retrieval is decisive when strategy text alone is a poor retrieval signal.

6.2 Impact of Strategy Dilution

A core motivation for selective curation is that the generator's performance degrades when its context is diluted with irrelevant strategies. We observe this effect empirically through three lenses.

Top-k versus probability threshold. The probability-threshold variant of DC-SCR adapts its retrieval cardinality based on the softmax score distribution: when scores are spread evenly, it retrieves many items to reach the threshold, flooding the generator with marginally relevant strategies. The 7.6 pp gap between the top-k ($k = 3$) and probability-threshold variant ($p = 0.8$) on AIME and 4.0 pp gap on IneqMath suggests that this indiscriminate retrieval actively harms reasoning.

Monolithic versus selective curation. The cumulative accuracy curves for IneqMath (Figure 3b) show that DC-Cu starts strong but gradually converges toward the Default as curation noise accumulates, while DC-DL maintains its advantage throughout the sequence. Together with DC-Cu's sub-baseline performance, these suggest the DC-Cu's curator over-generalizes specialized inequality techniques to accommodate the growing strategy set, due to having to rewrite the cheatsheet at every step.

Oracle-memory sensitivity analysis Our sensitivity analysis (Table 2) confirms this directly: injecting $n = 50$ distractors alongside a gold strategy degrades accuracy by 29.8%, establishing a concrete empirical upper bound on distractor tolerance.

6.3 When Does Structured Memory Help?

Our results reveal a clear boundary condition for the utility of strategy-based memory augmentation. Structured memory provides the largest gains on tasks with *high strategy transferability*: AIME (competition math with recurring techniques like modular arithmetic, IneqMath (inequality proofs with reusable bounding strategies), and MathEquationBalancer (systematic coefficient-solving procedures). On these tasks, DC-DL improves over the best baseline by 6.0–10.0 pp.

Conversely, on DataSIR, gains are more modest ($+4.0$ pp over Default), and DC-SCR actually degrades performance. This suggests that when problems primarily test pattern recognition over structured data rather than transferable reasoning strategies, the overhead of strategy extraction and curation outweighs its benefits unless dual-embedding retrieval (as in DC-DL) compensates.

We further evaluate on MMLU-Pro Engineering and Physics—knowledge-recall benchmarks where problems are largely independent—and find that our methods do not outperform DC baselines (Appendix C). This confirms that Dynamic Ledger is most effective when the task distribution contains recurring, transferable problem-solving patterns, and that adapting the framework to knowledge-centric tasks remains an open direction.

7 Conclusion

We introduced Strategic Chunk Retrieval and Dynamic Ledger, two extensions to the Dynamic Cheatsheet framework that replace monolithic memory with structured, chunk-level stores and selective curation. Dynamic Ledger further incorporates dual-embedding retrieval over both strategy text and source-problem embeddings, and uses explicit CRUD operations for fine-grained memory.

Across four benchmarks, Dynamic Ledger consistently achieves the best accuracy, with the largest gains on math-intensive tasks: $+6.0$ pp on AIME 2020–2024 and $+10.0$ pp on IneqMath over the strongest baselines from [1], and perfect accuracy on MathEquationBalancer. Our analysis reveals that these improvements stem from two complementary mechanisms: (1) selective curation that limits information loss to retrieved items, preventing strategy dilution as the store grows; and (2) dual-embedding retrieval that surfaces relevant strategies even when the current problem and stored entry share no surface-level similarity.

A clear boundary condition emerges from our MMLU-Pro evaluation (Appendix C): on knowledge-recall tasks where problems test factual memory rather than transferable strategies, the overhead of strategy extraction and curation outweighs its benefits, and simpler retrieval methods dominate. This points to an important open direction: automatically detecting low-transferability regimes at inference time and falling back to a lighter retrieval mode.

Finally, Confidence-Weighted Retrieval did not independently improve over DC baselines, likely because its trust signal is most valuable when the retrieved pool already contains high-quality candidates. However, it is architecturally orthogonal to both SCR and Dynamic Ledger—requiring no changes to memory structure or curation—making integration as a re-ranking layer a natural and promising next step.

In future work, we will use a stronger model for embedding or train an RL model to embed memories based on strategic similarities, since the current embedding model `text-embedding-3-small` only captures limited semantic similarity, reducing DL's capability. We will also improve memory storage efficiency (current DL has linear cost as shown in Appendix C).

Team Contributions

**All authors contributed equally; names in alphabetical order.*

Jerry Gu led the implementation of novel cheatsheet approaches, specifically Strategic Chunk Retrieval and Dynamic Ledger, as well as the project's fundraising efforts. Additionally, Jerry contributed to conducting experiments, analyzing results, adapting datasets, and writing the final report.

Shurui Liu led the execution of experiments, result analysis, codebase management, website development, and the writing of the final report. Shurui also led the formalization of DC using probability and information theory, and adaptation of the IneqMath and DataSIR datasets, while contributing to the Dynamic Ledger implementation and confidence score investigations.

Sabrina Yen-Ko led the ablation and sensitivity analysis, qualitative result analysis, and the implementation of confidence scores using ensembling methods. Sabrina also contributed to conducting experiments, analyzing results, and writing the final report.

Acknowledgment: The authors gratefully acknowledge the generous support of Y Combinator, OpenAI, and xAI. This research was made possible by $\$2,500$ in OpenAI credits provided through the YC AI Stack and $\$2,500$ in Grok API credits via xAI's YC Students Program.

References

[1] Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic Cheatsheet: Test-Time Learning with Adaptive Memory, April 2025. arXiv:2504.07952 [cs].

[2] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts, 2023.

[3] Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models, 2026.

[4] Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems, 2024.

[5] Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, 2025.

[6] Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *The Fourteenth International Conference on Learning Representations*, 2026.

[7] Qizheng Zhang, Michael Wornow, and Kunle Olukotun. Cost-efficient serving of LLM agents via test-time plan caching. In *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models*, 2025.

[8] Xiaofang Yang, Lijun Li, Heng Zhou, Tong Zhu, Xiaoye Qu, Yuchen Fan, Qianshan Wei, Rui Ye, Li Kang, Yiran Qin, Zhiqiang Kou, Daizong Liu, Qi Li, Ning Ding, Siheng Chen, and Jing Shao. Toward efficient agents: Memory, tool learning, and planning, 2026.

[9] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

[10] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[11] Joon Sung Park, Joseph C O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–22, 2023.

[12] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[13] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. GPT-4o system card, 2024.

[14] Jiayi Sheng, Luna Lyu, Jikai Jin, Tony Xia, Alex Gu, James Zou, and Pan Lu. Solving inequality proofs with large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

[15] Fan Mo, Bo Liu, Yuan Fan, Kun Qin, Yizhou Zhao, Jinhe Zhou, Jia Sun, Jinfei Liu, and Kui Ren. DataSIR: a benchmark dataset for sensitive information recognition. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

[16] Hamish Ivi. aime-2021-2025. `https://huggingface.co/datasets/hamishivi/aime-2021-2025`, 2025.

[17] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

Appendices

A Prompt Templates

All methods share a common generator–curator architecture. Below we describe the key structural differences between prompts; the full prompt text is available in our code repository.

A.1 Generator Prompt

The generator prompt is shared across all methods with minor wording changes. It instructs the model to (1) analyse the question alongside any provided strategies, (2) develop a step-by-step solution using Python for all numerical computation, and (3) output a final answer wrapped in `<answer>` tags. The prompt includes formatting rules for multiple-choice, numerical, and free-response answers.

For DC baselines, the context slot is labelled `CHEATSHEET` and filled with the curated cheatsheet text. For Dynamic Ledger, the slot is relabelled `RETRIEVED STRATEGIES` and filled with only the retrieved ledger entries, reinforcing that the context is a selective subset rather than a monolithic document.

A.2 Curator Prompts

The curator prompts differ substantially across methods, as they define the memory update mechanism.

DC-Cumulative / Strategic Chunk Retrieval. Both use the same prompt template. The curator receives the *previous cheatsheet* (DC-Cu: the full cheatsheet; DC-SCR: only the retrieved chunks), the current question, and the model's answer. It is instructed to output an updated cheatsheet wrapped in `<cheatsheet>` tags, consisting of `<memory_item>` blocks. Each block contains a `<description>` (problem context with reference tags such as `Q1`, `Q14`) and an `<example>` (code snippet or worked-out solution), followed by a usage counter. The prompt emphasizes that *any content not explicitly copied forward will be lost*, which is necessary because the curator produces a complete replacement cheatsheet. The cheatsheet is limited to approximately 2,000–2,500 words.

DC-Retrieval Synthesis. The curator receives the *retrieved past input–output pairs* (rather than a previous cheatsheet) plus the *next input* to solve. It synthesizes a query-specific cheatsheet from the retrieved examples, using the same `<memory_item>` output format. This prompt additionally instructs the curator to evaluate whether each retrieved solution was effective and to prioritize frequently useful strategies via a usage counter.

Dynamic Ledger. The curator prompt is fundamentally restructured around CRUD operations. It receives the *retrieved strategies* (displayed with their `unique_id`), the current question, and the model's answer. Instead of producing a full replacement cheatsheet, the curator outputs a JSON array of atomic operations wrapped in `<memory_updates>` tags:

```
<memory_updates>
[
  {"operation": "create",
   "strategy": "Full strategy text..."},
  {"operation": "update", "unique_id": 3,
   "strategy": "Revised strategy text..."},
  {"operation": "delete", "unique_id": 7}
]
</memory_updates>
```

Key design differences from DC-SCR:

- The curator must output *at least one* `create` or `update` operation per problem, biasing the system toward memory growth.
- Strategy text guidelines emphasize *technique-level* descriptions (e.g., "test the symmetric case") over problem-specific descriptions, improving retrieval generalization.

- The `delete` operation is reserved for demonstrably incorrect strategies, not for compression—the ledger size is managed by retrieval selectivity rather than aggressive pruning.
- Only retrieved entries (shown with their `unique_id`) can be updated or deleted; non-retrieved entries are invisible to the curator, providing a structural locality guarantee.

A.3 Ensemble Trust Scoring

Trust scoring adds three extra generation calls per problem using the same generator prompt but at temperatures $\{0.7, 0.8, 0.9\}$. Extracted answers are compared to the primary answer using the existing structured parser. The trust score s_i^{trust} is the proportion of agreement, and any example with $s_i^{\text{trust}} < 0.5$ is excluded from the retrieval candidate pool for all future problems.

B A Probabilistic Framework for Dynamic Cheatsheet

This appendix develops a formal probabilistic analysis of the DC framework and its variants. The central quantity is the conditional mutual information $I(\mathcal{K};\, y^* \mid x)$ between the knowledge state and the ground-truth answer: we show how each design choice—monolithic vs. selective curation, single- vs. dual-embedding retrieval—affects this quantity and thereby the achievable error rate.

B.1 Formal Setup and Notation

Task setting. Let $\mathcal{X}$ and $\mathcal{Y}$ be the input and output spaces. We fix a *task distribution* $\mathcal{D}$ over $(x, y^*) \in \mathcal{X} \times \mathcal{Y}$, where y^* is the ground-truth answer. A language model $\mathcal{M}$ defines a conditional distribution $P_{\mathcal{M}}(\hat{y} \mid \text{prompt})$ over outputs. At inference time the system receives queries $(x_1, x_2, \ldots, x_T)$ sequentially, where $x_t \sim \mathcal{D}_{\mathcal{X}}$ i.i.d.

Definition B.1 (Accuracy and Regret)**.** *The* per-step accuracy *given knowledge state $\mathcal{K}$ is* $\text{acc}(\mathcal{K}) \coloneqq \mathbb{P}_{\mathcal{M}}(\hat{y}_t = y_t^* \mid x_t,\, \mathcal{K})$*, the probability the generator answers correctly. The* average accuracy *and* cumulative regret *over T queries are*

$$\text{acc}_T = \frac{1}{T}\sum_{t=1}^{T} \mathbb{1}[\hat{y}_t = y_t^*], \qquad R_T = \sum_{t=1}^{T} \mathbb{1}[\hat{y}_t \neq y_t^*].$$

Note that $\mathbb{E}[\text{acc}_T] = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\text{acc}(\mathcal{K}_{t-1})]$.

Definition B.2 (Knowledge State)**.** *A* knowledge state *at step t is a string $\mathcal{K}_t \in \{0,1\}^*$ representing accumulated strategies. We say $\mathcal{K}_t$ is B-*bounded *if $|\mathcal{K}_t|_{\text{tok}} \leq B$.*

Definition B.3 (Curator as Markov Kernel)**.** *A* curator *$\mathcal{C}$ is a stochastic map producing an updated knowledge state: $\mathcal{K}_t \sim \mathcal{C}(\cdot \mid \mathcal{K}_{t-1},\, x_t,\, \hat{y}_t\,)$.*

Throughout, we assume the *generator Markov property*: $\hat{y}_t \perp y_t^* \mid (x_t,\, \mathcal{K}_{t-1})$, i.e. the prediction depends on the ground truth only through the knowledge state and the current input.

Let $\text{Enc} : \mathcal{X} \cup \{0,1\}^* \to \mathbb{R}^d$ be a fixed embedding function (in practice, `text-embedding-3-small`, $d = 1536$). Write $\text{Enc}_t \coloneqq \text{Enc}(x_t)$ and $\cos(u, v) \coloneqq u^\top v / (\|u\|\,\|v\|)$.

Definition B.4 (ϵ-Sufficient Statistic)**.** *A knowledge state $\mathcal{K}$ is an* ϵ-sufficient statistic *for predicting y^* given x (with respect to history $\mathcal{H}$) if $I(\mathcal{H};\, y^* \mid x) \leq I(\mathcal{K};\, y^* \mid x) + \epsilon$.*

B.2 Approach Models

Baseline. The baseline makes each prediction independently ($\mathcal{K}_t = \emptyset$ for all t), so $I(\mathcal{K}_{t-1}; y_t^* \mid x_t) = 0$. Any DC variant achieving positive conditional mutual information lowers the Fano error bound below the baseline level (Proposition B.2).

Full History Appending. The entire Q&A history $\mathcal{H}_{t-1} = \{(x_i, \hat{y}_i)\}_{i<t}$ is appended verbatim. This is feasible only for $t \leq T_{\max} \coloneqq \lfloor L/\bar{l} \rfloor$ (where L is the context window and $\bar{l}$ is the average Q&A token length). The fraction of relevant tokens (signal-noise-ratio) in the prompt is q for all t, where q is the task relevance fraction—constant and non-improving.

DC-Cumulative. Maintains a B-bounded knowledge state via iterative curator updates:

$$\hat{y}_t = \mathcal{M}(x_t, \mathcal{K}_{t-1}), \tag{4}$$

$$\mathcal{K}_t \sim \mathcal{C}(\cdot \mid \mathcal{K}_{t-1},\, x_t,\, \hat{y}_t). \tag{5}$$

The sequence $(\mathcal{K}_t)_{t\geq 0}$ is a time-inhomogeneous Markov chain.

Assumption B.1 (Curator Monotonicity)**.** *The curator $\mathcal{C}$ satisfies: for all t, $\mathbb{E}_{x_{t+1}}[I(\mathcal{K}_t;\, y^*_{t+1} \mid x_{t+1})] \geq \mathbb{E}_{x_{t+1}}[I(\mathcal{K}_{t-1};\, y^*_{t+1} \mid x_{t+1})]$.*

Proposition B.1 (Monotone Fano Bound)**.** *Under Assumption B.1, the Fano error lower bound for DC-Cumulative (Proposition B.2) is non-increasing in t: the information-theoretic floor on error shrinks monotonically, permitting progressively higher accuracy.*

Proof. By Proposition B.2, the error lower bound at step t is $\big(H(y^* \mid x) - \mathbb{E}_x[I(\mathcal{K}_{t-1}; y^* \mid x)] - 1\big)/\log|\mathcal{Y}|$. Assumption B.1 ensures $\mathbb{E}_x[I(\mathcal{K}_{t-1}; y^* \mid x)]$ is non-decreasing in t, so the error lower bound is non-increasing. □

Dynamic Retrieval. At step t, retrieve the top-k most similar past Q&A pairs: $\mathcal{R}_k(x_t) \coloneqq \arg\text{top-}\, k_{i<t}\, \cos(\mathrm{Enc}_t, \mathrm{Enc}_i)$. Under *perfect embedding separation* (relevant items rank above irrelevant ones), precision is $\mathrm{Prec}_k = \min(1, m_t/k)$, where m_t is the number of relevant past Q&A pairs for query x_t, and the relevance fraction satisfies $\mathrm{Prec}_k \geq q$, i.e. an improvement of factor $1/q$ over full history. Moreover, Dynamic Retrieval uses only $O(k\bar{l})$ tokens versus $O(t\bar{l})$ for Full History—a token-cost ratio of $(t-1)/k \to \infty$ for fixed k.

DC-Retrieval & Synthesis. After retrieval, a curator synthesises the raw pairs into a tailored cheatsheet $\mathcal{K}^{(t)}_{\mathrm{syn}}$. By the data processing inequality, $I(\mathcal{K}^{(t)}_{\mathrm{syn}}; y^*_t \mid x_t) \leq I(\mathcal{R}_k; y^*_t \mid x_t)$, but if $\mathcal{K}^{(t)}_{\mathrm{syn}}$ is an ϵ-sufficient statistic the loss is at most ϵ, while token cost drops from $O(k\bar{l})$ to $O(B)$.

DC-Strategic Chunk Retrieval. Maintains a structured memory store $\mathcal{S}_t = \{s_j\}_{j=1}^{N_t}$, where each chunk carries embedding $\mathrm{Enc}(s_j)$ and usage counter c_j. Retrieval uses the blended score (Eq. 1); the retrieved set $\mathcal{R}^*(x_t) \subseteq \mathcal{S}_{t-1}$ contains the top-κ chunks whose cumulative softmax probability exceeds threshold ρ (see Proposition B.4). After generating $\hat{y}_t$, only retrieved chunks are updated by the curator; non-retrieved items remain untouched:

$$\mathcal{S}_t = \big(\mathcal{S}_{t-1} \setminus \mathcal{R}_t\big) \cup \mathcal{C}(\mathcal{R}_t,\, x_t,\, \hat{y}_t), \tag{6}$$

where $\mathcal{C}$ denotes the curator producing updated versions of retrieved chunks.

DC-Dynamic Ledger. Maintains a structured store $\mathcal{S}_t = \{s_j\}_{j=1}^{N_t}$ where each entry carries a unique identifier id_j, strategy text, and *two* embeddings: a strategy embedding $\mathbf{v}_j = \mathrm{Enc}(\text{strategy}_j)$ and a source-problem embedding $\mathbf{u}_j = \mathrm{Enc}(\text{problem}_j)$. Retrieval operates over both embedding spaces simultaneously: given query x_t with embedding Enc_t, we retrieve the top-k items by each channel and take the union:

$$\mathcal{R}^{\mathrm{prob}}_t = \underset{j\leq N_{t-1}}{\arg\text{top-k}}\ \cos(\mathrm{Enc}_t,\, \mathbf{u}_j), \tag{7}$$

$$\mathcal{R}^{\mathrm{strat}}_t = \underset{j\leq N_{t-1}}{\arg\text{top-k}}\ \cos(\mathrm{Enc}_t,\, \mathbf{v}_j), \tag{8}$$

$$\mathcal{R}_t = \mathcal{R}^{\mathrm{prob}}_t \cup \mathcal{R}^{\mathrm{strat}}_t, \qquad k \leq |\mathcal{R}_t| \leq 2k. \tag{9}$$

After generation, the curator outputs CRUD (create/update/delete) operations. The store update takes the form:

$$\mathcal{S}_t = \big(\mathcal{S}_{t-1} \setminus (\mathcal{S}^-_t \cup \tilde{\mathcal{S}}_t)\big)\ \cup\ \mathcal{S}^{\mathrm{upd}}_t\ \cup\ \mathcal{S}^+_t, \tag{10}$$

where $\mathcal{S}^-_t$ denotes deleted entries, $\tilde{\mathcal{S}}_t \subset \mathcal{S}_{t-1}$ are the pre-update versions of revised items, $\mathcal{S}^{\mathrm{upd}}_t$ are their replacements (matched by identifier), and $\mathcal{S}^+_t$ are newly created entries. Updated and created items are re-embedded on both their strategy text and source problem. Unlike DC-SCR, the store size is variable: $N_t = N_{t-1} - |\mathcal{S}^-_t| + |\mathcal{S}^+_t|$.

B.3 Fano Bounds

Proposition B.2 (Fano Error Lower Bound)**.** *Let $\mathcal{K}$ be a knowledge state and $|\mathcal{Y}|$ finite. For any predictor based on $(x, \mathcal{K})$,*

$$\mathbb{P}(\hat{y} \neq y^* \mid x, \mathcal{K}) \;\geq\; \frac{H(y^* \mid x, \mathcal{K}) - 1}{\log |\mathcal{Y}|} \;=\; \frac{H(y^* \mid x) - I(\mathcal{K};\, y^* \mid x) - 1}{\log |\mathcal{Y}|}. \tag{11}$$

In particular, the Fano error bound is non-increasing in $I(\mathcal{K};\, y^* \mid x)$: higher mutual information permits lower error.

Proof. Apply Fano's inequality conditional on $(x, \mathcal{K})$: $H(y^* \mid \hat{y}, x, \mathcal{K}) \leq 1 + \mathbb{P}(\hat{y} \neq y^* \mid x, \mathcal{K}) \log |\mathcal{Y}|$. The generator Markov property $\hat{y} \perp y^* \mid (x, \mathcal{K})$ gives $H(y^* \mid \hat{y}, x, \mathcal{K}) = H(y^* \mid x, \mathcal{K})$. Substituting, rearranging, and using $H(y^* \mid x, \mathcal{K}) = H(y^* \mid x) - I(\mathcal{K}; y^* \mid x)$ yields (11). □

Theorem B.1 (Rate-Distortion Lower Bound)**.** *Let $R^* = I(\mathcal{H}_{t-1};\, y_t^* \mid x_t)$. For any cheatsheet $\mathcal{K}$ with distortion $D(\mathcal{K}; \mathcal{H}_{t-1} \mid x_t) \coloneqq R^* - I(\mathcal{K};\, y_t^* \mid x_t) \leq \delta$,*

$$|\mathcal{K}|_{\text{tok}} \;\geq\; \frac{R^* - \delta}{H_{\text{tok}}},$$

where H_{tok} is the per-token entropy of the cheatsheet encoding.

Proof. $I(\mathcal{K}; y_t^* \mid x_t) \geq R^* - \delta$ by the distortion bound. Since $I(\mathcal{K}; y_t^* \mid x_t) \leq H(\mathcal{K} \mid x_t) \leq H(\mathcal{K})$ and $H(\mathcal{K}) \leq H_{\text{tok}} \cdot |\mathcal{K}|_{\text{tok}}$ (each of $|\mathcal{K}|_{\text{tok}}$ tokens carries at most H_{tok} nats), the result follows. □

Theorem B.1 establishes a fundamental token-length lower bound: no approach can represent the full task history in fewer than $(R^* - \delta)/H_{\text{tok}}$ tokens while maintaining distortion $\leq \delta$.

B.4 Regret Analysis

Assumption B.2 (Two-Phase Accuracy)**.** *There exists a saturation step m such that for $t \leq m$, $\mathbb{E}[\text{acc}(\mathcal{K}_{t-1})] = p_0$ (baseline accuracy), and for $t > m$, $\mathbb{E}[\text{acc}(\mathcal{K}_{t-1})] = p_1 > p_0$.*

Theorem B.2 (Cumulative Regret Improvement)**.** *Under Assumption B.2 with m deterministic,*

$$\mathbb{E}[R_T^{\text{base}} - R_T^{\text{DC}}] = (T - m)(p_1 - p_0) \geq 0,$$

and $\mathbb{E}[\text{acc}_T^{\text{DC}}] - \mathbb{E}[\text{acc}_T^{\text{base}}] = \frac{T-m}{T}(p_1 - p_0) \to p_1 - p_0$ as $T \to \infty$.

Proof. $\mathbb{E}[R_T^{\text{base}}] = T(1-p_0)$. Under the two-phase model, $\mathbb{E}[R_T^{\text{DC}}] = m(1-p_0) + (T-m)(1-p_1)$. Subtracting: $\mathbb{E}[R_T^{\text{base}} - R_T^{\text{DC}}] = (T-m)[(1-p_0) - (1-p_1)] = (T-m)(p_1 - p_0)$. □

B.5 Convergence of the Knowledge Chain

Theorem B.3 (Strategy Coverage)**.** *Suppose $\mathcal{D}$ is generated by K distinct latent strategies, each occurring with equal probability $1/K$. The curator captures a new strategy with conditional probability $p > 0$. Let T^* be the first step at which all K strategies are in the cheatsheet. Then $\mathbb{E}[T^*] = (KH_K)/p = O(K \log K/p)$, where H_K is the K-th harmonic number. After $T > 2K \ln K/p$ queries, the cheatsheet is complete with probability $\geq 1 - 1/K$.*

Proof. We use a coupon-collector argument. Define *phase* k ($k = 1, \ldots, K$) as the period during which exactly $k-1$ strategies have been captured and we await the k-th. In phase k the probability of capturing a new strategy per step is $\pi_k = p(K - k + 1)/K$, so the expected phase length is $\mathbb{E}[\ell_k] = 1/\pi_k = K/[p(K - k + 1)]$. Summing: $\mathbb{E}[T^*] = \sum_{k=1}^{K} \mathbb{E}[\ell_k] = (K/p) \sum_{j=1}^{K} 1/j = KH_K/p$. The tail bound follows from the classical coupon-collector concentration bound $\mathbb{P}(T^* > K(\ln K + c)/p) \leq e^{-c}$; setting $e^{-c} = 1/K$ gives $c = \ln K$. □

B.6 Retrieval Quality Analysis: Signal-Noise-Ratio

For a query x_t, let $\mathcal{P}_t \subseteq \mathcal{S}_{t-1}$ denote the set of items whose strategy content is semantically relevant to solving x_t.

Proposition B.3 (Frequency Bonus Cannot Flip Rankings)**.** *Define the* blended score $\sigma(s_j, x_t) \coloneqq \alpha \cos(\mathrm{Enc}_t, \mathrm{Enc}(s_j)) + (1-\alpha) f(c_j)$, *where* $f(c_j) = \ln(1+c_j)/\ln(1+c_{\max}) \in [0,1]$ *is the normalised log-frequency and* $\alpha \in (0,1]$. *Let* $\Delta = \min_{j\in\mathcal{P}_t} \cos(\mathrm{Enc}_t, \mathrm{Enc}(s_j)) - \max_{j\notin\mathcal{P}_t} \cos(\mathrm{Enc}_t, \mathrm{Enc}(s_j)) > 0$ *be the embedding margin. If* $(1-\alpha)\sup_j f(c_j) < \alpha\Delta$, *every relevant item outranks every irrelevant item under* σ, *so the top-*k *set for* $k \le |\mathcal{P}_t|$ *consists entirely of relevant items. At* $\alpha = 0.85$, *this requires* $\Delta > 3/17 \approx 0.176$.

Proof. For any relevant chunk $j^* \in \mathcal{P}_t$ and irrelevant $j' \notin \mathcal{P}_t$: $\sigma(s_{j^*}, x_t) - \sigma(s_{j'}, x_t) \ge \alpha\Delta - (1-\alpha)\sup_j f(c_j) > 0$ under the stated condition, so relevant items always outrank irrelevant ones. □

Proposition B.4 (Adaptive Cardinality under Majorization)**.** *For a probability distribution* $\mathbf{p}$ *over scores, define the* retrieval cardinality $\kappa(\mathbf{p}) \coloneqq \min\{k : \sum_{j=1}^{k} p_j \ge \rho\}$, *where* $\rho \in (0,1)$ *is a fixed probability threshold. Let* $\mathbf{p}$ *and* $\mathbf{p}'$ *be the softmax distributions over chunk scores for two queries, both sorted in decreasing order. If* $\mathbf{p}$ *majorises* $\mathbf{p}'$ *(i.e.* $\sum_{j=1}^{k} p_j \ge \sum_{j=1}^{k} p'_j$ *for all* k*), then* $\kappa(\mathbf{p}) \le \kappa(\mathbf{p}')$*: the more concentrated distribution retrieves fewer chunks.*

Proof. Since $\mathbf{p}$ majorises $\mathbf{p}'$, $\sum_{j=1}^{k} p_j \ge \sum_{j=1}^{k} p'_j$ for all k. The cumulative sum of $\mathbf{p}$ therefore crosses the threshold ρ at a smaller or equal index than that of $\mathbf{p}'$, giving $\kappa(\mathbf{p}) \le \kappa(\mathbf{p}')$. □

Proposition B.5 (Locality of Memory Update)**.** *Under DC-Strategic Chunk Retrieval, the fraction of the store modified at step* t *is* $|\mathcal{R}^*(x_t)|/N_t$. *For fixed probability threshold* ρ *and* $N_t \to \infty$, *this fraction* $\to 0$.

Proof. Exactly $|\mathcal{R}^*(x_t)|$ chunks are replaced at step t. In the geometric score-gap model (Corollary B.1), κ depends on τ, Δ, ρ but not on N_t, so $\kappa/N_t \to 0$. □

Corollary B.1 (Geometric Model: Exact Cardinality)**.** *Suppose scores follow a geometric gap structure* $\sigma_j = \sigma_1 - (j-1)\Delta$ *with softmax temperature* $\tau > 0$ *(so that* $p_j \propto e^{\sigma_j/\tau}$*). Then* $p_j = (1-r)r^{j-1}$ *with* $r = e^{-\Delta/\tau}$ *and* $\kappa = \lceil -\log(1-\rho)\,\tau/\Delta \rceil$. *In the two limiting cases:* $\kappa \to N$ *as* $\Delta \to 0$ *(uniform scores) and* $\kappa \to 1$ *as* $\Delta \to \infty$ *(concentrated scores).*

Proposition B.6 (Dynamic Ledger Recall Dominance)**.** *The recall of DC-Dynamic Ledger retrieval* $\mathcal{R}_t$ *(Eq. 9) satisfies*

$$\mathrm{Rec}(\mathcal{R}_t) \;\ge\; \max\big(\mathrm{Rec}(\mathcal{R}_t^{\mathrm{prob}}),\; \mathrm{Rec}(\mathcal{R}_t^{\mathrm{strat}})\big),$$

where $\mathrm{Rec}(\mathcal{R}) \coloneqq |\mathcal{R} \cap \mathcal{P}_t| \,/\, |\mathcal{P}_t|$.

Proof. By construction $\mathcal{R}_t \supseteq \mathcal{R}_t^{\mathrm{prob}}$ and $\mathcal{R}_t \supseteq \mathcal{R}_t^{\mathrm{strat}}$, so $\mathcal{R}_t \cap \mathcal{P}_t \supseteq (\mathcal{R}_t^{\mathrm{prob}} \cap \mathcal{P}_t) \cup (\mathcal{R}_t^{\mathrm{strat}} \cap \mathcal{P}_t)$, giving $|\mathcal{R}_t \cap \mathcal{P}_t| \ge \max\big(|\mathcal{R}_t^{\mathrm{prob}} \cap \mathcal{P}_t|,\, |\mathcal{R}_t^{\mathrm{strat}} \cap \mathcal{P}_t|\big)$. Dividing by $|\mathcal{P}_t|$ yields the result. □

Proposition B.7 (Dynamic Ledger MI Gain)**.** *Under DC-Dynamic Ledger retrieval,*

$$I(\mathcal{R}_t;\, y_t^* \mid x_t) \;\ge\; I(\mathcal{R}_t^{\mathrm{strat}};\, y_t^* \mid x_t).$$

In particular, equality holds when $\mathcal{R}_t^{\mathrm{prob}} \subseteq \mathcal{R}_t^{\mathrm{strat}}$.

Proof. Since $\mathcal{R}_t^{\mathrm{strat}} \subseteq \mathcal{R}_t$ by construction, monotonicity of conditional entropy gives $H(y_t^* \mid x_t, \mathcal{R}_t) \le H(y_t^* \mid x_t, \mathcal{R}_t^{\mathrm{strat}})$. Subtracting both sides from $H(y_t^* \mid x_t)$ yields the inequality. When $\mathcal{R}_t^{\mathrm{prob}} \subseteq \mathcal{R}_t^{\mathrm{strat}}$, $\mathcal{R}_t = \mathcal{R}_t^{\mathrm{strat}}$ and equality is immediate. □

Assumption B.3 (Faithful Curation)**.** *The curator's CRUD operations at step* t *satisfy:*

(i) ***Safe deletion:*** *the deleted items are jointly conditionally redundant given the remaining store:* $I(\mathcal{S}_t^-;\, y^* \mid x,\, \mathcal{S}_{t-1} \setminus \mathcal{S}_t^-) = 0$ *for* $\mathcal{D}$*-a.e.* (x, y^*).

(ii) ***Non-destructive update:*** *the store after deletions and in-place updates (but before creates), $\mathcal{S}_t^{\text{mid}} := (\mathcal{S}_{t-1} \setminus (\mathcal{S}_t^- \cup \tilde{\mathcal{S}}_t)) \cup \mathcal{S}_t^{\text{upd}}$, satisfies $I(\mathcal{S}_t^{\text{mid}}; y^* \mid x) \geq I(\mathcal{S}_{t-1} \setminus \mathcal{S}_t^-; y^* \mid x)$ for $\mathcal{D}$-a.e. (x, y^*).*

Proposition B.8 (CRUD Store Monotonicity)**.** *Under Assumption B.3,*

$$\mathbb{E}_{x_{t+1}}\big[I(\mathcal{S}_t; y_{t+1}^* \mid x_{t+1})\big] \;\geq\; \mathbb{E}_{x_{t+1}}\big[I(\mathcal{S}_{t-1}; y_{t+1}^* \mid x_{t+1})\big].$$

Proof. Let $A = \mathcal{S}_{t-1} \setminus \mathcal{S}_t^-$. By the chain rule, $I(\mathcal{S}_{t-1}; y^* \mid x) = I(A; y^* \mid x) + I(\mathcal{S}_t^-; y^* \mid x, A)$. Condition (i) gives $I(\mathcal{S}_t^-; y^* \mid x, A) = 0$, so $I(A; y^* \mid x) = I(\mathcal{S}_{t-1}; y^* \mid x)$. Condition (ii) gives $I(\mathcal{S}_t^{\text{mid}}; y^* \mid x) \geq I(A; y^* \mid x)$. Finally, $I(\mathcal{S}_t; y^* \mid x) = I(\mathcal{S}_t^{\text{mid}}; y^* \mid x) + I(\mathcal{S}_t^+; y^* \mid x, \mathcal{S}_t^{\text{mid}}) \geq I(\mathcal{S}_t^{\text{mid}}; y^* \mid x)$ by non-negativity of conditional MI. Chaining the three bounds and taking expectations over x_{t+1} completes the proof. □

B.7 Mechanisms of Improvement

The preceding results characterize each approach in isolation. We now explain *why* selective curation (DC-SCR and DC-DL) outperforms monolithic curation (DC-Cu) and why DC-Dynamic Ledger extends the gains of DC-SCR.

The monolithic-curation bottleneck. DC-Cu rewrites the entire cheatsheet at every step. As the store grows to N_t strategies, three mutually reinforcing failure modes arise: *(a) strategy dilution*—the curator must reconcile all strategies in a single pass, over-generalizing task-specific strategies; *(b) irrelevant-context contamination*—the generator's prompt includes all strategies, and the "lost-in-the-middle" effect [2] attenuates information from strategies buried deep in the context; *(c) error propagation*—a single bad curation step corrupts the entire cheatsheet, and all subsequent steps inherit the damage. Selective curation addresses all three by restricting both the generator and curator to $|\mathcal{R}_t| \ll N_t$ retrieved items.

Proposition B.9 (Non-Retrieved Item Preservation)**.** *Under selective curation (Eq. 6 for DC-SCR, Eq. 10 for DC-DL), every non-retrieved item $s \in \mathcal{S}_{t-1} \setminus \mathcal{R}_t$ is carried forward to $\mathcal{S}_t$ unchanged. Writing $s^{(t)}$ for the copy of s in $\mathcal{S}_t$,*

$$I\big(s^{(t)}; y^* \mid x\big) \;=\; I\big(s^{(t-1)}; y^* \mid x\big) \quad \textit{for all } (x, y^*) \sim \mathcal{D}.$$

Under monolithic curation, every strategy passes through the curator channel at every step; by the data-processing inequality, each such pass can only preserve or lose information about y^*, never create it.

Proof. By construction (Eqs. 6, 10), items outside $\mathcal{R}_t$ (and outside $\mathcal{S}_t^-$ for DC-DL) are carried forward as identical random variables, so their MI with any external quantity is unchanged. □

Assumption B.4 (Curator Capacity)**.** *When the curator processes n items simultaneously, each item incurs expected information loss $\epsilon(n) \geq 0$, where ϵ is non-decreasing in n (more items ⇒ less attention per item).*

Proposition B.10 (Cumulative Curation Loss)**.** *Under Assumption B.4, with store size N and retrieval cardinality κ held constant, the total information loss per strategy item j over T curation steps satisfies:*

$$\textit{Monolithic:} \quad L_j^{\text{mono}} = T \cdot \epsilon(N), \tag{12}$$

$$\textit{Selective:} \quad L_j^{\text{sel}} = n_j \cdot \epsilon(\kappa), \tag{13}$$

where $n_j \leq T$ is the number of times item j is retrieved and $\kappa \ll N$. For any item with $n_j < T$, the savings are twofold: fewer curation exposures ($n_j < T$) and lower per-exposure loss ($\epsilon(\kappa) \leq \epsilon(N)$).

Proof. Under monolithic curation, item j passes through the curator at every step (T times), each time alongside all N strategies, incurring loss $\epsilon(N)$ per pass. Under selective curation, item j is curated only when retrieved (n_j times), each time alongside at most $\kappa - 1$ other items, incurring loss $\epsilon(\kappa)$ per pass; by Proposition B.9, the remaining $T - n_j$ steps contribute zero loss. □

Together with Proposition B.5 (the modified fraction vanishes as $N_t \to \infty$), this shows that selective curation achieves bounded total distortion even as the strategy store grows without limit.

Dynamic Ledger retrieval complementarity. Propositions B.6 and B.7 show that DC-Dynamic Ledger retrieval achieves recall and MI at least as high as either single embedding channel. The strict gain arises when the two channels retrieve *different* relevant items: the problem channel surfaces strategies whose source problem resembles the current query (high $\cos(\mathrm{Enc}_t, \mathbf{u}_j)$), while the strategy channel surfaces strategies whose technique is applicable regardless of surface similarity (high $\cos(\mathrm{Enc}_t, \mathbf{v}_j)$). This is particularly valuable for *transfer* across structurally dissimilar problems that share underlying techniques—precisely the setting where DC-DL shows its largest empirical gains on math benchmarks.

B.8 Information-Theoretic Comparison of All Approaches

Table 3 summarises the mutual information between the generator's context and y^*, together with token cost and relevance (signal-noise-ratio), for each approach. The key insight is the *information–token trade-off*: Full History maximises MI but at unbounded token cost; DC-Cu compresses to $O(B)$ tokens but risks curation loss that grows with the store (Section B.7); DC-SCR and DC-DL achieve scalable token cost $O(\kappa\bar{s})$ or $O(2k\bar{s})$ while preserving non-retrieved strategies exactly (Proposition B.9), keeping cumulative curation loss bounded (Proposition B.10).

Table 3: Mutual information, token cost, and scalability comparison across DC variants. Here $\bar{l}$ is the average Q&A token length and $\bar{s}$ is the average strategy-chunk token length ($\bar{s} \ll \bar{l}$ in practice).

Approach	**MI** $I(\cdot\,; y^* \mid x)$	**Tokens**	**Scales?**	**Signal-Noise-Ratio**
Baseline	0	$O(1)$	Yes	0
Full History	$I(\mathcal{H}_{t-1}; y^* \mid x)$	$O(t\bar{l})$	No	q
DC-Cumulative	$\geq I(\mathcal{H}_{t-1}; y^* \mid x) - \epsilon$	$O(B)$	Yes	$\geq q - \epsilon/B$
Dynamic Retrieval	$I(\mathcal{R}_k; y^* \mid x)$	$O(k\bar{l})$	Yes	Prec_k
DC-RS	$\geq I(\mathcal{R}_k; y^* \mid x) - \epsilon$	$O(B)$	Yes	$\geq \mathrm{Prec}_k - \epsilon/B$
DC-SCR (ours)	$I(\mathcal{R}^*; y^* \mid x)$	$O(\kappa\bar{s})$	Yes	Prec_κ
DC-DL (ours)	$I(\mathcal{R}_t; y^* \mid x)$	$O(2k\bar{s})$	Yes	$\mathrm{Prec}_{\lvert\mathcal{R}_t\rvert}$

Proposition B.11 (Ordering of Mutual Information)**.** *For t sufficiently large and under perfect embedding separation:*

$$0 = I(\emptyset;\, y^* \mid x) \leq I(\mathcal{R}_k;\, y^* \mid x) \leq I(\mathcal{H}_{t-1};\, y^* \mid x).$$

Moreover, if the curator is ϵ-sufficient (Definition B.4), then after saturation step T^* the cheatsheet satisfies $I(\mathcal{K}_{t-1};\, y^* \mid x) \geq I(\mathcal{H}_{t-1};\, y^* \mid x) - \epsilon \geq I(\mathcal{R}_k;\, y^* \mid x) - \epsilon$. Within the memory-store approaches, DC-Dynamic Ledger satisfies $I(\mathcal{R}_t;\, y^* \mid x) \geq I(\mathcal{R}_t^{\mathrm{strat}};\, y^* \mid x)$ by Proposition B.7.

Proof. $I(\emptyset; y^* \mid x) = 0$ since the empty set carries no information. $I(\mathcal{R}_k; y^* \mid x) \geq 0$ by non-negativity of MI. $I(\mathcal{R}_k; y^* \mid x) \leq I(\mathcal{H}_{t-1}; y^* \mid x)$ by monotonicity of mutual information under set inclusion ($\mathcal{R}_k \subseteq \mathcal{H}_{t-1}$). After T^*, $\mathcal{K}_{t-1}$ is an ϵ-sufficient statistic for $\mathcal{H}_{t-1}$ (Definition B.4), giving the first bound; the second follows from $\mathcal{R}_k \subseteq \mathcal{H}_{t-1}$. □

C Limitation Analysis on Knowledge-Recall Tasks: MMLU Pro Benchmarks

The benchmarks in Section 5 were selected to stress-test strategy reuse on *reasoning-heavy* tasks where problems share underlying techniques. To probe the limits of this assumption, we additionally evaluate on two MMLU-Pro [17] subsets—**Engineering** and **Physics**—each containing 250 multiple-choice questions drawn from graduate-level professional exams. Unlike AIME or IneqMath, these questions primarily test domain knowledge and factual recall rather than transferable problem-solving strategies.

C.1 Results

Table 4 and Figures 4–6 present the full results. Three patterns stand out:

Table 4: Accuracy (%) on MMLU-Pro subsets (250 questions each). Methods above the mid-rule are baselines from [1]; methods below are ours. † indicates the result is below the Default baseline.

	Engineering		Physics	
Method	GPT-4o	GPT-5	GPT-4o	GPT-5
Default / Baseline	53.6	64.8	76.0	83.2
EmptyCheatsheet	52.8	—	75.2	—
DC_Cumulative	46.1	63.6	76.0	85.2
FullHistoryAppend	—	72.0	—	89.6
Dynamic_Retrieval	48.8	72.0	—	89.6
DC_RetrievalSynth	51.6	**72.4**	75.6	**89.2**
DC_StrategicChunk (ours)	53.6	66.8	73.2†	82.8†
DC_DynamicLedger (ours)	51.6	67.2	72.0†	85.2

Our methods do not outperform DC baselines. On GPT-5 Engineering, DC_RetrievalSynth reaches 72.4% while DC_DynamicLedger achieves only 67.2% and DC_StrategicChunk 66.8%. On GPT-5 Physics, DC_StrategicChunk (82.8%) falls *below* the Default baseline (83.2%), and DC_DynamicLedger merely matches DC_Cumulative at 85.2%. The same pattern holds on GPT-4o, where DC_DynamicLedger drops to 72.0% on Physics versus the 76.0% Default.

Simple retrieval methods dominate. FullHistoryAppend and Dynamic_Retrieval (72.0% / 89.6% on GPT-5) outperform all structured-memory methods. These approaches inject raw past question–answer pairs without curation, suggesting that on knowledge-recall tasks the model benefits more from seeing verbatim examples than from distilled strategies.

Memory overhead without accuracy gains. Figure 6 shows that DC_StrategicChunk accumulates over 1,200 KB of memory on both subsets, and DC_DynamicLedger over 700 KB, while DC baselines remain below 50 KB. This storage growth yields no accuracy benefit, representing wasted context budget and increased latency.

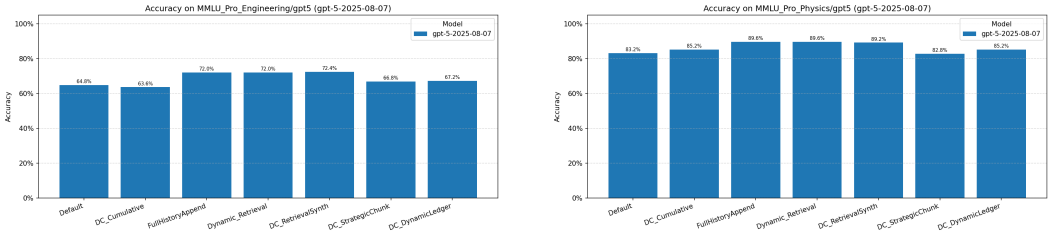
C.2 Discussion

We attribute the underperformance to a fundamental mismatch between our method's inductive bias and the task structure of MMLU-Pro:

1. **Low strategy transferability.** MMLU-Pro questions test domain-specific knowledge (e.g., Faraday's law parameters, thermodynamic cycle efficiencies) rather than reusable problem-solving strategies. Two questions on electromagnetic induction may share no common "strategy" beyond knowing Maxwell's equations—knowledge the model either already possesses or cannot acquire from a single example. The curator therefore generates vague, overly broad strategies that provide little actionable guidance.
2. **Strategy noise in multiple-choice settings.** In free-response math benchmarks, correct strategies encode genuine procedural knowledge (e.g., "reduce modular exponentiation via Euler's theorem"). In multiple-choice settings, the "strategy" extracted from a correct answer is often little more than the answer itself or a surface-level rationale that does not generalize. Retrieving such entries for a new question adds noise without informational value.
3. **High baseline competence.** GPT-5 already achieves 83.2% on Physics without any memory augmentation. At this accuracy level, most remaining errors are due to genuine knowledge gaps or ambiguous questions rather than missing strategies. Additional context from a strategy store is unlikely to fill factual gaps and may instead introduce distracting information, consistent with the "lost in the middle" effect [2].
4. **Topic heterogeneity.** MMLU-Pro Engineering and Physics span dozens of sub-topics (circuits, thermodynamics, optics, mechanics, etc.) with little overlap between consecutive questions. The embedding-based retrieval therefore rarely finds genuinely relevant past entries, and the retrieved strategies act as distractors rather than aids.

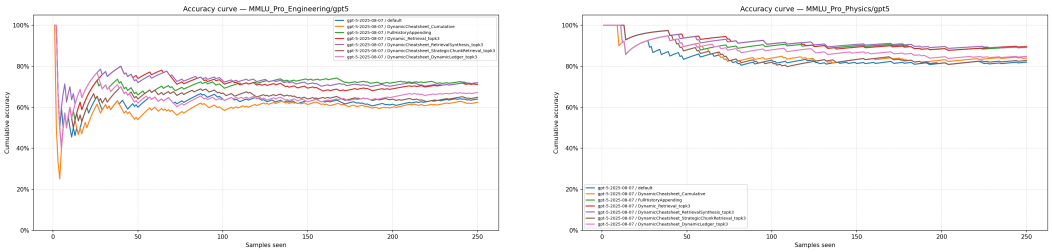
These results highlight an important boundary condition: Dynamic Ledger and Strategic Chunk Retrieval are most effective when the task distribution contains recurring, transferable problem-solving patterns. On knowledge-recall benchmarks where problems are largely independent and factual, the overhead of strategy extraction and curation outweighs its benefits. Adapting the framework to such settings—for example, by caching factual associations rather than procedural strategies, or by detecting low transferability and falling back to a simpler retrieval mode—remains an open direction for future work.

C.3 MMLU-Pro Figures



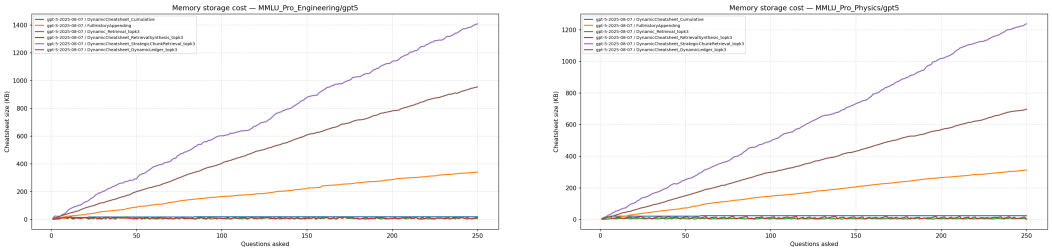
(a) MMLU-Pro Engineering (GPT-5) (b) MMLU-Pro Physics (GPT-5)

Figure 4: Accuracy comparison on MMLU-Pro subsets. Unlike the math benchmarks (Figure 2), DC_StrategicChunk and DC_DynamicLedger do not outperform the DC baselines.



(a) Engineering: cumulative accuracy (b) Physics: cumulative accuracy

Figure 5: Cumulative accuracy curves on MMLU-Pro (GPT-5). On Physics, all methods gradually decline from near-perfect early accuracy, with DC_StrategicChunk consistently tracking below the Default baseline.



(a) Engineering: memory cost (b) Physics: memory cost

Figure 6: Memory storage cost on MMLU-Pro (GPT-5). DC_StrategicChunk and DC_DynamicLedger accumulate significantly more memory than baselines, yet this additional storage does not translate into accuracy gains.