REMIND: Enhancing CoT Reliability with Self-Supervised Introspection

Souvik Maji*
Indian Institute of Technology Jodhpur
Jodhpur, Rajasthan, India
majisouvik2022@gmail.com

Dibyanayan Bandyopadhyay*
Indian Institute of Technology Patna
Patna, Bihar, India
dibyanayan@gmail.com

Asif Ekbal
Indian Institute of Technology Patna
Patna, Bihar, India
asif@iitp.ac.in

Abstract

Chain-of-Thought (CoT) prompting has significantly enhanced the reasoning capabilities of Large Language Models (LLMs). However, CoT can also produce fluent but logically flawed reasoning chains that make incorrect answers appear more reliable. In this work, we introduce **REMIND** (**R**eliability **E**nhancement via **M**etacognitive **IN**trospection and **D**ebiasing), a framework that enables LLMs to learn from their own failures via episodic memory. Unlike standard error-correction methods that rely on external feedback or extensive fine-tuning, REMIND leverages a self-supervised introspection cycle. We first isolate epistemic failure—instances where CoT decreases model confidence despite yielding incorrect results. The model then performs structured introspection to diagnose the root cause and synthesizes a reusable natural language lesson. These lessons are stored in a vector database and retrieved during future inference steps to correct similar future errors. Experiments on eight diverse reasoning benchmarks spanning mathematical, scientific, multi-hop, truthfulness, and commonsense reasoning demonstrate that REMIND significantly improves reliability by effectively converting past CoT failures into actionable insights for unseen problems. Codes are available in https://github.com/newcodevelop/REMIND.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in complex reasoning tasks, particularly when prompted with Chain-of-Thought (CoT) strategies (Wei et al., 2022). By decomposing problems into intermediate steps, CoT allows models to tackle multi-hop reasoning challenges in mathematics, commonsense, and symbolic logic. However, a prevalent limitation is the static nature of standard inference. When an LLM makes a mistake, it typically lacks the mechanism to reflect on that failure, extract a lesson, and apply it to future scenarios. This deficiency is particularly acute in CoT failures where the generation of a fluent but flawed reasoning chain enhances the model's false notion of certainty, a phenomenon known as the reasoning illusion or faithfulness gap (Turpin et al., 2023).

Human reasoning, by contrast, is inherently metacognitive and episodic. When a human expert errs, they engage in *introspection*: they analyze their reasoning trace to identify the specific logic error or overlooked constraint, generalize this into a "lesson" (e.g., "When seeing variable X, I must check for condition Y"), and store this in episodic memory. Upon encountering a similar problem later, this memory is triggered, preempting the repetition of the same error.

In this work, we propose **REMIND** (**R**eliability **E**nhancement via **M**etacognitive **IN**trospection and **D**ebiasing), a novel framework that endows LLMs with a similar capability to learn from their own reasoning failures without requiring weight updates. *Our approach operates on the insight that LLMs possess latent self-correction abilities that are rarely activated during standard generation (either with or without CoT).*

The REMIND methodology proceeds in three distinct phases. First, during a "practice" phase, the model attempts to solve problems using both standard and CoT prompting. We systematically categorize outputs based on correctness and confidence, isolating a subset of *epistemic failures*, cases where the explicit reasoning path misleads the model into an incorrect answer, where the model is also uncertain. Second, we trigger an *introspection module* where the model is prompted to analyze its reasoning trajectory. It identifies the root cause (e.g., logical fallacy, missed constraint, calculation error) and synthesizes a reusable natural language

*Souvik Maji and Dibyanayan Bandyopadhyay contributed equally to this work.

Proceedings of the 15th International Joint Conference on Natural Language Processing and the 5th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 2917–2942
November 6-10, 2026 ©2026 Association for Computational Linguistics

"lesson". These lessons are encoded and stored in an external vector memory, effectively building a library of "anti-patterns" and warnings.

REMIND transfers answer-free failure lessons rather than solved examples, and any ground-truth signal is used only during offline memory construction; a fuller deployment and memorization discussion appears in Appendix A.9 and A.10.

Finally, during inference on new instances, the model queries this introspection memory. By retrieving lessons from semantically similar past failures, the model augments its context with specific warnings (e.g., "Warning: deeply verify constraints regarding integer division"). This memory-augmented context guides the CoT generation, allowing the model to navigate around pitfalls it has previously encountered. A comparison between direct inference, CoT and our methodology (REMIND) is shown in Figure 1.

Our contributions are as follows:

(i) We introduce the **REMIND** framework, a training-free architecture that enables LLMs to improve reasoning reliability through self-generated introspection and episodic memory.

(ii) We propose a novel taxonomy for failure detection based on confidence-reasoning mismatches, and analyse association with model confidence calibration and performance.

(iii) We demonstrate that augmenting CoT with retrieved introspections improves performance on eight diverse benchmarks spanning mathematical, scientific, multi-hop, truthfulness, and commonsense reasoning, converting past failures into future successes.

(iv) We provide ablations and analyses that isolate the source of the gains: memory composition, retrieval depth, retrieval strategy, memory-size scaling, cross-model transfer, online inference cost, symbolic failure-type shifts, and human judgments of generated lessons.

2 Related Work

2.1 Faithfulness and Reliability in Chain-of-Thought Reasoning

The introduction of Chain-of-Thought (CoT) prompting has significantly advanced the reasoning capabilities of Large Language Models (LLMs) by decomposing complex tasks into intermediate steps (Wei et al., 2022). However, recent studies have highlighted a critical "faithfulness gap" where the generated reasoning chain does not accurately reflect the model's underlying decision process (Turpin et al., 2023; Lanham et al., 2023). Turpin et al. (2023) demonstrated that models often generate post-hoc rationalizations for incorrect answers, creating an "illusion of reasoning" that can mislead users. This phenomenon is exacerbated by the model's tendency towards overconfidence, where fluent but logically flawed explanations are assigned high probability scores (Huang et al., 2024). While previous works have attempted to mitigate this via consistency checks (Wang et al., 2023) or external verifiers (Cobbe et al., 2021), they typically treat reasoning errors as stochastic noise rather than systematic failures. In contrast, our work specifically targets these "epistemic failures" as training signals, leveraging the mismatch between confidence and accuracy to trigger metacognitive introspection.

2.2 Self-Correction and Introspection in LLMs

Prior work studies self-correction through iterative self-feedback, such as *Reflexion* (Shinn et al., 2023) and *Self-Refine* (Madaan et al., 2023). However, evidence suggests that purely intrinsic self-correction is often weak once a model has committed to an erroneous answer, especially without external feedback (Huang et al., 2024). Our approach instead separates *introspection* from inference: rather than repairing an answer online, the model analyzes a confirmed failure offline to identify its root cause and distill a reusable lesson. This framing is closer to metacognitive monitoring, which has been argued to support more effective downstream control (Wang and Zhao, 2024).

2.3 Mistake Learning and Reasoning-Trained Models

REMIND is related to recent mistake-learning methods such as LeMa and LEMMA, which also argue that model errors contain useful supervision signals (An et al., 2024; Pan et al., 2025). The broad philosophy of learning from mistakes is not new. The distinction is the operating regime: LeMa and LEMMA convert mistakes into correction or revision trajectories and internalize them through fine-tuning, whereas REMIND keeps the base model frozen, stores abstract structural/procedural lessons externally, and retrieves them at inference time. Thus, REMIND is not intended to replace fine-tuning-based mistake learning; it targets deployments where parameter access, retraining com-

Question (MATH dataset)

Problem: A Senate committee has 8 Republicans and 6 Democrats. In how many ways can we form a subcommittee of 5 members with at least one member from each party? **Ground Truth:** 1940

Baseline (Direct)

Model Output:
ANS = 4510

Analysis:

- No reasoning.
- Hallucinated answer.
- **Accuracy: 0%**

Standard CoT

Reasoning Trace:

1. $4R, 1D$: $\binom{8}{4}\binom{6}{1} = 420$
2. $3R, 2D$: $\binom{8}{3}\binom{6}{2} = 840$
3. $2R, 3D$: $\binom{8}{2}\binom{6}{3} = 560$

Total = 1820
ANS = 1820 (×)

Failure: Missed case $1R, 4D$ ($= 120$).

REMIND + CoT (Ours)

1. Total: $\binom{14}{5} = 2002$
2. All Republicans: $\binom{8}{5} = 56$
3. All Democrats: $\binom{6}{5} = 6$

$2002 - 56 - 6 = 1940$
ANS = 1940 (✓)

Why it works: Retrieved reasoning tips and records of prior similar errors (e.g., avoiding missing cases) from memory.

Figure 1: **Qualitative comparison on MATH (Hendrycks et al., 2021).** Standard CoT fails due to missing case enumeration, while REMIND retrieves a complementary counting strategy.

1. Failure Detection
Compare direct decoding and CoT on training examples. Keep cases where

$$r_{\text{CoT}} > r_{\text{direct}} \quad \text{and} \quad \hat{y}_{\text{CoT}} \neq y.$$

Output: failure episodes

→

2. Memory Construction
For each failure, ask the model what went wrong, what pattern caused it, and how to avoid it next time.

Store the resulting lesson in an episodic memory bank.

Output: reusable failure memory

→

3. Memory-Augmented Inference
Given a new query, retrieve relevant lessons and prepend them to the prompt.

$$x_{\text{new}} \rightarrow \text{retrieve} \rightarrow \text{augment} \rightarrow \hat{y}$$

Output: more reliable reasoning

Figure 2: **High-level overview of REMIND.** The framework first identifies cases where CoT produces underconfident wrong answers, then converts those failures into reusable lessons, and finally retrieves relevant lessons at test time to guide reasoning on new queries.

pute, or model-specific repeated fine-tuning is impractical.

REMIND is also complementary to reasoning-tuned or "thinking" models. Training-time reasoning optimization can internalize self-verification behaviors, but deployed models can still repeat systematic errors on new inputs. REMIND provides a complementary post-hoc reliability layer: it lets a frozen model accumulate an external memory of prior reasoning traps without modifying its weights.

2.4 Memory-Augmented Generation for Reasoning

Prior memory-augmented generation methods typically retrieve factual knowledge, similar examples, or past feedback to support reasoning (Lewis et al., 2020; Madaan et al., 2022; Modarressi et al., 2024). In contrast, REMIND indexes *failure lessons*: reusable warnings about common reasoning pitfalls rather than problem instances themselves. By retrieving such lessons, our method emphasizes debiasing through cautionary guidance, placing it at the intersection of episodic memory for generation and safety-aware reasoning (Zhang and Choi, 2023; Huet et al., 2025).

Novelty. Unlike previous works that rely on heavy fine-tuning or human-in-the-loop feedback, and agentic methodology, REMIND establishes a self-supervised loop where the model functions as its own critic and memory curator, specifically optimizing for the reuse of underconfident failure lessons using chain of thought (CoT).

We evaluate the memory architecture through comparisons with MemPrompt and RAG and through ablations of episodic memory, the global constitution, retrieval strategy, retrieval depth, and memory size. These analyses show that the gains arise from relevant failure lessons and the combi-

nation of local and global memory.

3 Methodology

We propose **REMIND** (Reliability Enhancement via Metacognitive INtrospection and Debiasing), a neuro-symbolic framework designed to mitigate reasoning hallucinations in Large Language Models. Our approach operates on the hypothesis that reasoning failures stem from two distinct sources: (1) *Structural Isomorphisms*, where specific mathematical patterns trigger consistent cognitive traps; and (2) *Procedural Drift*, where the model violates general meta-cognitive heuristics (e.g., failing to check constraints).

To address this, REMIND implements a **Dual-Stream Architecture** comprising four phases: (1) *Entropy-Based Failure Detection*; (2) *Dual-Layer Introspection*; (3) *Knowledge Consolidation*; and (4) *Constitution-Guided Inference*. Figure 2 shows a high-level diagram of the proposed methodology, while we posit the full diagram of REMIND in Appendix Figure 4.

3.1 Failure Detection via Entropy Dynamics

We identify *underconfident failures*, incorrect answers where CoT reasoning increases uncertainty, as primary indicators of *epistemic* gaps solvable via knowledge injection. We quantify sequence uncertainty r using mean token-level Shannon entropy:

$$r(x, \hat{y}) = \frac{1}{T} \sum_{t=1}^{T} H(P_t(\cdot \mid x, \hat{y}_{<t})) \qquad (1)$$

We isolate failure cases where CoT amplifies uncertainty relative to direct generation ($\Delta r = r_{\text{CoT}} - r_{\text{direct}} > 0$), serving as high-value signals of the model's inherent logical uncertainty.

Terminology is fixed as follows throughout the paper. An *underconfident failure* is an incorrect CoT output with $\Delta r > 0$, meaning that explicit reasoning increases uncertainty. An *overconfident failure* is an incorrect output with $\Delta r < 0$, meaning that CoT decreases uncertainty despite being wrong. We use *epistemic failure* as the broader category of incorrect examples whose confidence dynamics expose a mismatch between reasoning and correctness; the main REMIND memory is constructed from the underconfident subset because these cases empirically show the strongest receptivity to retrieved guidance.

3.2 Dual-Layer Metacognitive Introspection

Upon detecting a confirmed failure tuple $(x, \hat{y}_{\text{CoT}}, y)$, where y is the ground-truth answer and $\hat{y}_{\text{CoT}} \neq y$, we trigger a specialized introspection process. Unlike prior works that generate unstructured feedback, our method prompts the model to decompose the error into two orthogonal layers $D = (L_{struct}, L_{proc})$:

Layer 1: Structural Lesson (L_{struct}). This layer captures the specific mathematical constructs that triggered the failure in the first place. The model isolates: i) **Abstract Trigger**: The mathematical configuration (e.g., "Polynomials with even powers") stripped of specific numbers, ii) **Cognitive Trap**: The specific heuristic abuse (e.g., "Assumed linearity in non-linear functions"), iii) **Structure Fix**: A conditional logic rule: *IF [Trigger] THEN [Verification Step].*

Layer 2: Procedural Flaw (L_{proc}). This layer identifies the meta-cognitive "bad habits" that allowed the error to propagate, which are globally aggregated later. The model outputs a **Process Tag** (e.g., "Constraint Blindness") and a universal **Process Rule** (e.g., "Always verify domain constraints before simplifying"). The prompt to construct L_{struct} and L_{proc} is defined in Figure 11.

Because self-diagnosis can be noisy, REMIND does not assume that every individual introspection is correct. Noise is mitigated through three mechanisms: (i) lessons are induced only from confirmed offline failures, (ii) structural and procedural lessons are stored separately, allowing local traps and global rules to reinforce each other, and (iii) retrieval aggregates over semantically similar failures rather than relying on a single raw explanation. We therefore evaluate lesson quality explicitly in Appendix A.4.

3.3 Knowledge Consolidation

The extracted layers are processed into two distinct knowledge stores:

3.3.1 Episodic Failure Memory (EFM)

We utilize the L_{struct} components to populate a vector database using `FAISS`. We encode the concatenation of the trigger and trap description using a pre-trained sentence transformer (`all-MiniLM-L6-v2`). This ensures that future retrieval is driven by the *structural mechanics* of the problem rather than surface-level lexical overlap.

3.3.2 Global Reasoning Constitution

While EFM handles specific traps, the Global Constitution addresses systemic deficits. We aggregate all L_{proc} entries and perform a *Frequency-Based Pruning and Semantic Consolidation*. The specific prompt is shown in Figure 12.

We calculate the frequency distribution of Process Tags across the failure set. The most frequent failure modes are synthesized by the same LLM into a concise Reasoning Constitution (C_{global}), a static set of high-priority "universal laws" that the LLM must obey. Figure 5 demonstrates one such global constitution (C_{global}).

3.4 Constitution-Guided Inference

During the inference phase for a new question x_{new}, we employ a memory-augmented generation pipeline that dynamically integrates both knowledge streams. The process follows three steps:

1. Dual-Stream Context Construction. We retrieve the top-k (specifically $k = 3$) most relevant failure episodes $M_{retrieved}$ from the EFM based on cosine similarity with x_{new}. $M_{retrieved}$ represents the serialized content of the top-k retrieved failure episodes, serving as dynamic, context-aware guardrails. Simultaneously, we load the static Global Constitution C_{global}.

2. Prompt Assembly. The system prompt is constructed by hierarchically layering the general rules and specific retrieval contexts before the operational directives:

$$\mathcal{P} = [C_{global}] \oplus [M_{retrieved}] \oplus [\mathcal{I}_{meta}] \oplus [x_{new}] \quad (2)$$

Where: $\mathcal{I}_{meta}$ is the **Instruction Block**. It acts as a standard operating procedure containing three parts: (1) A command to apply the retrieved memories (e.g., *"CHECK against the specific failure patterns listed if applicable (You may or may not apply)"*); (2) A solved example (few shot) to demonstrate the reasoning style (e.g., CoT: *"Question: Janet has 16 apples... ANS = 12"*); and (3) Strict output rules to ensure the answer is readable (e.g., *"Response MUST end with EXACTLY: ANS = <value>"*).

This structure ensures that the model's attention is primed with both global principles and specific historical pitfalls before it attempts to solve the new problem.

3. Constrained Generation. To ensure the retrieved knowledge is utilized, the model is prompted to output a "Self-Correction" block before the final answer. The instructions (shown in Figure 13) explicitly constrain the model to: (1) Check if the current problem matches the retrieved warning signs in $M_{retrieved}$, and (2) Apply the verification steps mandated by C_{global}. This effectively primes the attention mechanism to attend to potential pitfalls before generating the reasoning chain.

4 Results and Analysis

In this section, we evaluate the REMIND framework across a diverse set of eight reasoning benchmarks spanning mathematical, scientific, and commonsense reasoning tasks. The evaluated datasets include **GSM8K** (Cobbe et al., 2021), **MATH** (Hendrycks et al., 2021), **GPQA** (Rein et al., 2024), **ARC-C** (Clark et al., 2018), **HotpotQA** (Yang et al., 2018), **Minerva** (Lewkowycz et al., 2022), **TruthfulQA** (Lin et al., 2022), and **StrategyQA** (Geva et al., 2021). Table 1 summarizes the aggregate accuracy and predictive entropy across all datasets.

Our primary objective is to evaluate whether REMIND improves reasoning reliability by reducing reasoning failures and producing more calibrated predictions. In addition to standard accuracy, we analyze predictive entropy as a measure of reasoning stability and confidence. We further investigate the effect of memory construction strategies and identify which types of failure cases provide the most effective signals for episodic learning.

4.1 Experimental Setup

All experiments use the **Qwen2.5-7B** model (Qwen et al., 2025) as the base language model. Evaluation is conducted on the standard test splits of each dataset.

For constructing the *Episodic Failure Memory*, we collect diverse reasoning examples separately from the evaluation data to avoid any leakage. For mathematical reasoning benchmarks (GSM8K, MATH, and Minerva), memory examples are drawn from the **NuminaMath-CoT** dataset.[1] Specifically, we sample a diverse set of 100 reasoning problems to populate the episodic memory used during inference.

For non-mathematical reasoning benchmarks

[1] https://huggingface.co/datasets/AI-MO/NuminaMath-CoT

(ARC-C, HotpotQA, GPQA, TruthfulQA, and StrategyQA), we similarly construct the memory using 100 examples sampled from the respective datasets, ensuring that these examples do not overlap with the evaluation instances.

REMIND, RAG, and MemPrompt all use the same 100 labeled practice examples. RAG retrieves the raw solved examples, while REMIND transforms confirmed failures into answer-free, structured lessons capturing reusable reasoning patterns and verification strategies. Thus, REMIND's gains over RAG isolate the benefit of converting examples into introspective lessons, rather than access to additional labeled data.

This setup ensures that episodic memory construction remains independent from the test set while providing a diverse set of failure patterns for introspective lesson extraction. Additional implementation details and hyperparameter settings are provided in Appendix A.5.

Ground-truth labels are used only to select failed practice examples during offline memory construction, never at inference. We also include a label-free self-consistency pseudo-label variant and additional cross-model experiments; full details appear in Appendix A.9 and A.10.2.

4.2 Main Results

Table 1 shows a clear and highly consistent outcome: REMIND is the best-performing method across all eight benchmarks. It achieves the highest accuracy and the lowest predictive entropy in every dataset, outperforming not only the standard CoT baseline but also stronger inference-time baselines such as Self-Refine (Madaan et al., 2023), Self-Consistency (Wang et al., 2023), MemPrompt (Madaan et al., 2022), Meta-Prompting (Suzgun and Kalai, 2024), and RAG. This is the central empirical result of the paper.

The improvement is not confined to one task family. REMIND remains strongest on mathematical reasoning benchmarks such as GSM8K, MATH, and Minerva; on knowledge-intensive and scientific reasoning datasets such as ARC-C and GPQA; and on multi-hop, commonsense, and truthfulness-oriented tasks such as HotpotQA, StrategyQA, and TruthfulQA. This breadth is important because it shows that the method is not exploiting quirks of a single benchmark, but providing a general improvement in reasoning performance.

Equally important, the gains in accuracy are accompanied by systematic reductions in entropy. Thus, REMIND does not merely make the model more confident in an arbitrary sense; it makes the model both *more correct* and *more decisive*. The same pattern holds uniformly across all datasets, suggesting that the retrieved episodic lessons are helping the model avoid repeated reasoning failures while also stabilizing its inference behavior.

Among the baselines, Self-Consistency is often the strongest competitor. REMIND nevertheless obtains a higher accuracy point estimate and lower predictive entropy on each of the eight evaluated benchmarks. This indicates that the benefit of REMIND is not simply due to additional sampling or iterative refinement. Instead, the advantage appears to come from its core mechanism: converting prior failures into reusable introspective memories that can guide future reasoning.

REMIND and Self-Consistency operate on different components of the inference procedure. REMIND modifies the reasoning context through retrieved failure lessons, whereas Self-Consistency improves robustness by aggregating multiple sampled generations. The two mechanisms are therefore conceptually complementary and could in principle be combined. Evaluating such a combined system is outside the scope of the present experiments, which compare REMIND and Self-Consistency under their respective standalone configurations.

More importantly, Self-Consistency incurs substantial decoding and token costs because it generates 40 full reasoning paths for every example, whereas REMIND requires only one standard CoT generation, a lightweight embedding/FAISS lookup, and a fixed memory of 100 examples. Although the accuracy margins over Self-Consistency are modest, REMIND achieves these gains with one standard CoT generation, a lightweight embedding/FAISS lookup, and a fixed memory of 100 examples, whereas Self-Consistency uses 40 full reasoning paths per example, making REMIND more practical for real-world deployment.

The gains are not explained by prompt length alone: the dual-memory, retrieval-strategy, and retrieval-depth ablations in Appendix Section A.2 show that relevant failure lessons, rather than arbitrary added context, drive the improvement. In online decoding cost, REMIND requires one full generation plus a lightweight embedding/FAISS lookup, whereas Self-Consistency and Self-Refine require multiple LLM generations; the detailed cost comparison is in Table 9.

Configuration	Accuracy (↑)	Entropy (↓)	Configuration	Accuracy (↑)	Entropy (↓)
Dataset: GSM8K			*Dataset: ARC-C*		
Baseline (CoT)	0.854 ±0.015	0.110 ±0.003	Baseline (CoT)	0.637 ±0.017	0.119 ±0.004
Self-Refine	0.863 ±0.014	0.098 ±0.003	Self-Refine	0.645 ±0.016	0.112 ±0.004
Self-Consistency	0.869 ±0.013	0.089 ±0.003	Self-Consistency	0.651 ±0.016	0.105 ±0.004
MemPrompt	0.854 ±0.015	0.110 ±0.002	MemPrompt	0.640 ±0.017	0.118 ±0.005
Meta-Prompting	0.861 ±0.014	0.101 ±0.003	Meta-Prompting	0.643 ±0.016	0.113 ±0.004
RAG	0.857 ±0.015	0.107 ±0.003	RAG	0.639 ±0.017	0.117 ±0.004
REMIND	**0.879** ±0.012	**0.056** ±0.002	**REMIND**	**0.662** ±0.015	**0.078** ±0.003
Dataset: MATH			*Dataset: HotpotQA*		
Baseline (CoT)	0.780 ±0.039	0.118 ±0.005	Baseline (CoT)	0.211 ±0.019	0.151 ±0.007
Self-Refine	0.791 ±0.038	0.103 ±0.005	Self-Refine	0.215 ±0.018	0.146 ±0.006
Self-Consistency	0.795 ±0.038	0.097 ±0.004	Self-Consistency	0.215 ±0.018	0.139 ±0.006
MemPrompt	0.784 ±0.040	0.115 ±0.003	MemPrompt	0.213 ±0.020	0.150 ±0.006
Meta-Prompting	0.789 ±0.038	0.106 ±0.005	Meta-Prompting	0.214 ±0.018	0.147 ±0.006
RAG	0.783 ±0.039	0.115 ±0.005	RAG	0.212 ±0.019	0.149 ±0.007
REMIND	**0.805** ±0.037	**0.052** ±0.003	**REMIND**	**0.220** ±0.017	**0.110** ±0.005
Dataset: GPQA			*Dataset: Minerva*		
Baseline (CoT)	0.364 ±0.020	0.145 ±0.006	Baseline (CoT)	0.093 ±0.015	0.172 ±0.010
Self-Refine	0.379 ±0.019	0.132 ±0.005	Self-Refine	0.113 ±0.014	0.158 ±0.009
Self-Consistency	0.387 ±0.018	0.121 ±0.005	Self-Consistency	0.113 ±0.014	0.147 ±0.009
MemPrompt	0.368 ±0.021	0.143 ±0.006	MemPrompt	0.096 ±0.015	0.170 ±0.010
Meta-Prompting	0.376 ±0.019	0.134 ±0.005	Meta-Prompting	0.110 ±0.014	0.160 ±0.009
RAG	0.367 ±0.020	0.143 ±0.006	RAG	0.096 ±0.015	0.169 ±0.010
REMIND	**0.411** ±0.017	**0.094** ±0.004	**REMIND**	**0.121** ±0.013	**0.125** ±0.007
Dataset: TruthfulQA			*Dataset: StrategyQA*		
Baseline (CoT)	0.564 ±0.018	0.121 ±0.004	Baseline (CoT)	0.750 ±0.016	0.104 ±0.004
Self-Refine	0.564 ±0.017	0.116 ±0.004	Self-Refine	0.755 ±0.015	0.097 ±0.004
Self-Consistency	0.571 ±0.017	0.109 ±0.004	Self-Consistency	0.758 ±0.015	0.090 ±0.004
MemPrompt	0.561 ±0.018	0.123 ±0.004	MemPrompt	0.748 ±0.016	0.106 ±0.004
Meta-Prompting	0.564 ±0.017	0.117 ±0.004	Meta-Prompting	0.753 ±0.015	0.099 ±0.004
RAG	0.563 ±0.018	0.120 ±0.004	RAG	0.752 ±0.016	0.102 ±0.004
REMIND	**0.587** ±0.016	**0.082** ±0.003	**REMIND**	**0.763** ±0.014	**0.070** ±0.003

Table 1: **Aggregate Performance Across Eight Benchmarks.** Accuracy and entropy comparisons across datasets. Values report mean ± 95% confidence interval. Self-Consistency is evaluated using 40 independently sampled reasoning trajectories from the same LLM. Here, accuracy denotes final-answer correctness under each benchmark's evaluation protocol. Entropy is the mean token-level Shannon entropy over the full generated sequence, averaged across benchmark examples, rather than entropy only on the final answer token.

Overall, the main takeaway from Table 1 is straightforward: REMIND consistently improves reasoning quality across diverse benchmarks, and it does so in a way that is simultaneously more accurate and lower in uncertainty. This makes the empirical evidence for the framework both strong and unusually uniform.

As additional benchmark-level evidence, REMIND has a higher accuracy point estimate than Self-Consistency on all eight datasets, with an average difference of 1.11 accuracy points and consistently lower predictive entropy. An exact two-sided sign test over the eight benchmark-level comparisons gives $p = 0.0078$, indicating that the uniform direction of the benchmark-level differences is unlikely under an equal win/loss hypothesis. This analysis concerns the direction of benchmark-level comparisons and should not be interpreted as an instance-level significance test for individual datasets.

4.3 Optimality of Memory Formation

A critical question in designing self-correcting agents is: *Which failure modes constitute the most effective learning signals?* We posit a theoretical framework for *Epistemic Receptivity* to explain memory optimality.

We define the optimality of a memory set $\mathcal{M}$ based on its ability to maximize the model's likelihood of self-correction, denoted as $P(\text{correct}|\mathcal{M})$. We hypothesize that not all errors are equal. Specifically, we categorize failures based on the model's confidence during the erroneous generation:

i) Overconfident Failures (Delusion): The model is wrong but highly confident ($\Delta r = r_{\text{CoT}} - r_{\text{direct}} < 0$). This implies a strong internal misalignment or "knowledge hallucination."

ii) Underconfident Failures (Uncertainty):

The model is wrong and exhibits low confidence ($\Delta r = r_{\text{CoT}} - r_{\text{direct}} > 0$). This implies a recognizable "knowledge gap" or uncertainty.

Hypothesis. We argue that *Underconfident* samples are the optimal seeds for memory $\mathcal{M}$. If an LLM is overconfident about a mistake, it effectively "believes" it is correct; retrieving a correction for such a case often results in the model ignoring the memory due to its own high internal prior. Conversely, underconfident errors represent a *zone of epistemic uncertainty* where the model is uncertain and thus statistically more receptive to external guidance (the retrieved memory). This interpretation also clarifies an important failure mode of memory-augmented inference: there are cases where a relevant lesson may be retrieved but not acted upon because the model's internal prior remains too strong. The weaker performance of memory built from overconfident failures in Table 2 is consistent with this mechanism, suggesting that high-prior errors are less receptive to corrective retrieval even when an appropriate warning is available.

To validate this hypothesis, we performed an ablation study constructing $\mathcal{M}$ using strictly Overconfident, Random, or Underconfident failure samples. We evaluate this strategy across several reasoning benchmarks including GSM8K, MATH, GPQA, and StrategyQA.

Dataset	Overconfident	Random	Underconfident
GSM8K	0.866	0.872	**0.879**
MATH	0.796	0.801	**0.805**
GPQA	0.398	0.405	**0.411**
StrategyQA	0.758	0.760	**0.763**

Table 2: **Ablation on Memory Composition across datasets.** Accuracy obtained when episodic memory is constructed using different failure selection strategies. Underconfident failures consistently produce the strongest performance, supporting the Epistemic Receptivity hypothesis.

Table 2 shows that the **Underconfident** strategy consistently performs best across datasets, yielding the highest accuracy in each case. Random sampling provides moderate gains, while Overconfident failures are least effective as memory seeds. This pattern supports the intuition that failures occurring under uncertainty are more amenable to correction through retrieved lessons.

We performed ten independent random selections and reported the best-performing result in Table 2 rather than the average. This gives the Random baseline a favorable comparison, but Underconfident memory still performs better on all four datasets. Random memory may also contain some Underconfident failures, which explains why its performance generally lies between the Underconfident and Overconfident variants.

Underconfident memory achieves the highest accuracy point estimate on all four benchmarks in Table 2, with a statistically significant improvement over Random memory on GSM8K. A paired McNemar test for Underconfident versus Random memory on GSM8K is reported in Appendix A.10.1.

A paired McNemar test further supports this trend for Underconfident versus Random memory on GSM8K; details are provided in Appendix A.10.1.

Consequently, the optimal memory $\mathcal{M}^*$ should prioritize failures where the model's confidence is uncertain rather than strongly miscalibrated. See Figure 3

4.4 Sensitivity to Model Architecture

REMIND is not tied to Qwen2.5-7B. On a difficult MATH subset, it improves Llama-3.2-3B-Instruct from 45% to 53%, Phi-3.5-mini-instruct from 49% to 56%, and Qwen2.5-7B-Instruct from 77% to 81%. A further Llama-3.2-8B evaluation improves accuracy and reduces entropy on all eight benchmarks, including ARC-C, HotpotQA, TruthfulQA, and StrategyQA (Tables 10–11).

4.5 Process Fidelity and Error Recoverability

Accuracy alone hides failure severity: REMIND more often turns early cognitive collapse into structurally aligned near-misses that are easier to verify or repair. The human lesson-quality study, and symbolic-checker analysis are provided in Appendices A.4 and A.3.

5 Conclusion

We introduced REMIND, a training-free framework that improves LLM reasoning reliability by turning past CoT failures into reusable introspective memories. REMIND identifies uncertain failure cases, distills them into structured lessons, and retrieves them to guide future reasoning. Across eight benchmarks, it improves accuracy and predictive entropy over standard CoT and stronger inference-time baselines, with underconfident failures providing the most effective memories. Be-

Problem (MATH Dataset): The circle $x^2 + y^2 = 2$ and the parabola $y^2 = 8x$ have two common tangents, forming four points of tangency. Find the area of the quadrilateral formed by the four points of tangency.

Ground Truth: 15

Baseline: Cognitive Collapse (Schema Error)

Model Output Excerpt:

"Substitute $y^2 = 8x$ into circle... $x^2 + 8x - 2 = 0$
... Coordinates approx $(0.34, 2.8)$
... Area calculation ... **ANS = 8**"

Metrics: Token Count: 702 **Status:** Unrecoverable

Failure Analysis:

- **Wrong Logic:** Used *Intersection* ($f(x) = g(x)$) instead of *Tangency*.
- **Consequence:** Derived phantom points where curves cross, not touch.

REMIND (Ours): Structural Alignment (Execution Slip)

Model Output Excerpt:

"Condition: Dist from center = radius ($d = \sqrt{2}$)...
Solved $m = \pm 1$. Points: $(2, \pm 4), (-1, \pm 1)$...
Shape is a Rectangle (8×3)... **ANS = 24**"

Metrics: Token Count: 918 **Status:** Recoverable

Success Analysis:

- **Correct Logic:** Used $d = r$ formula; solved hard algebra perfectly.
- **Error:** *Visual Hallucination* (Trapezoid $\rightarrow$ Rectangle) at the very last step.

Detailed Process Verification

Criterion	Baseline (Standard CoT)	REMIND (Ours)
1. Understanding (Common Tangents)	× Incorrect *Failed to grasp problem structure.*	✓ Correct *Correctly identified goal.*
2. Methodology	× Wrong (Curve Intersection) *Equated $f(x) = g(x)$ directly.*	✓ Correct (True Tangents) *Used distance formula condition.*
3. Tangent Equations	× Not valid *No slopes derived.*	✓ Correct *Derived $m = \pm 1$.*
4. Points of Tangency	× Incorrect *Found intersection points.*	✓ Correct *Found all 4 coordinates.*
5. Geometric Interpretation	× Fabricated Geometry *Hallucinated non-existent shape.*	× Rectangle Assumption *Misidentified Trapezoid as Rectangle.*
6. Area Computation	× Inconsistent *Math did not follow from vertices.*	× Wrong Method *Valid calculation for the wrong shape.*
7. Logical Consistency	× Self-contradictory	✓ Mostly Consistent
OVERALL VERDICT	**Zero Credit**	**Partial Credit**

Figure 3: **Process Fidelity Analysis.** On a hard geometry problem, both methods miss the ground-truth answer (15), but their failures differ: REMIND (right) follows the correct tangency-based setup and fails only at the final shape classification, whereas the baseline (left) makes a fundamental logic error early. Full outputs appear in Figures 8 and 9 in Appendix A.13.

yond final-answer gains, REMIND also shifts residual errors toward more structurally aligned reasoning, making failures easier to verify or repair. Overall, REMIND shows that explicit memory of introspected failures can make LLM reasoning more reliable, calibrated, and tool-ready without parameter updates.

Limitations

While REMIND demonstrates promising results, several limitations warrant discussion.

Dependency on Oracle Feedback. The current failure detection module relies on access to ground truth labels to confirm when a generated practice answer is incorrect. This is an offline memory-construction requirement, not a test-time requirement: once the memory is built, REMIND only retrieves lessons and prompts the model. In real-world deployment scenarios where ground truth is unavailable, the system would need to rely on unsupervised uncertainty quantification or consistency checks (e.g., self-consistency voting) to identify potential failures for introspection or may rely on synthetic datasets created by more potent LLMs. Our self-consistency pseudo-label experiment suggests that such label-free alternatives are feasible but noisy, motivating future verifier- or execution-feedback-based memory mining.

Black-Box Log-Probability Access. REMIND uses token-level probabilities to mine underconfident failures during offline memory construction. The inference-time pipeline itself only needs retrieval and prompting, so it applies directly to API models that expose log-probabilities. For strict black-box models that provide no probability information, entropy-based failure mining

must be replaced by proxy signals such as self-consistency disagreement, answer instability across paraphrases, external verifier scores, execution feedback, or user corrections.

Retrieval Latency and Context Window. Augmenting the prompt with retrieved lessons inevitably increases the input token count, which linearly scales inference cost and latency. While we limited retrieval to the top-k lessons ($k = 3$), scaling the episodic memory to thousands of diverse failure modes may require more sophisticated selection mechanisms to prevent context window saturation or "lost-in-the-middle" effects where retrieved warnings are ignored. In our setup, however, the dominant online cost remains a single autoregressive generation; Table 9 summarizes this cost relative to Self-Consistency, Self-Refine, RAG, and MemPrompt.

Single-Turn Introspection. Our current introspection module operates in a single turn. If the model's self-diagnosis is flawed (e.g., it attributes a calculation error to a logic error), the stored lesson will be noisy. Future iterations could employ a multi-turn verification step where the model must prove its diagnosis is correct before storage.

Acknowledgements

Dibyanayan Bandyopadhyay gratefully acknowledges support from the Prime Minister's Research Fellowship (PMRF) under Grant No. 2703564. The authors also gratefully acknowledge the Ministry of Electronics and Information Technology (MeitY), Government of India, for supporting this work under the BHASHINI initiative through the COIL-D: Centre of Indian Language Data (Project No. SP/CSE/CIL/2024-25/1155). The authors also thank the anonymous reviewers for their constructive feedback and valuable suggestions.

References

Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. 2024. Learning from mistakes makes llm better reasoner. *Preprint*, arXiv:2310.20689.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *Preprint*, arXiv:1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Yu, Xinying Song, and Denny Zhou. 2024. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations*, volume 2024, pages 32808–32824.

Alexis Huet, Zied Houidi, and Dario Rossi. 2025. Episodic memories generation and evaluation benchmark for large language models. In *International Conference on Learning Representations*, volume 2025, pages 51409–51457.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, and 11 others. 2023. Measuring faithfulness in chain-of-thought reasoning. *Preprint*, arXiv:2307.13702.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 3843–3857. Curran Associates, Inc.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human

falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. 2022. Memory-assisted prompt editing to improve GPT-3 after deployment. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2833–2861, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pages 46534–46594. Curran Associates, Inc.

Ali Modarressi, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schuetze. 2024. RET-LLM: Towards a General Read-Write Memory for Large Language Models. In *ICLR 2024 Workshops: AGI*.

Zhuoshi Pan, Yu Li, Honglin Lin, Qizhi Pei, Zinan Tang, Wei Wu, Chenlin Ming, H. Vicky Zhao, Conghui He, and Lijun Wu. 2025. LEMMA: Learning from errors for MatheMatical advancement in LLMs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11615–11639, Vienna, Austria. Association for Computational Linguistics.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

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*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652. Curran Associates, Inc.

Mirac Suzgun and Adam Tauman Kalai. 2024. Meta-prompting: Enhancing language models with task-agnostic scaffolding. *Preprint*, arXiv:2401.12954.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. 2023. Language models don't always say what they think: unfaithful explanations in chain-of-thought prompting. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Yuqing Wang and Yun Zhao. 2024. Metacognitive prompting improves understanding in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1914–1926, Mexico City, Mexico. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed 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*, volume 35, pages 24824–24837. Curran Associates, Inc.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Michael Zhang and Eunsol Choi. 2023. Mitigating temporal misalignment by discarding outdated facts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14213–14226, Singapore. Association for Computational Linguistics.

A Appendix

A.1 Memory Pruning

To control memory growth and remove redundant lessons, REMIND applies pruning mechanisms to both L1 and L2 memories.

A.1.1 L1 Episodic Memory Pruning

Episodic lessons are clustered in embedding space and near-duplicate lessons are removed. Additionally, lessons that are rarely retrieved during inference may be pruned.

In practice, episodic memory contains substantial local redundancy because multiple failures often yield semantically similar warnings. We therefore cluster lessons in embedding space and remove near-duplicates before indexing, while also pruning

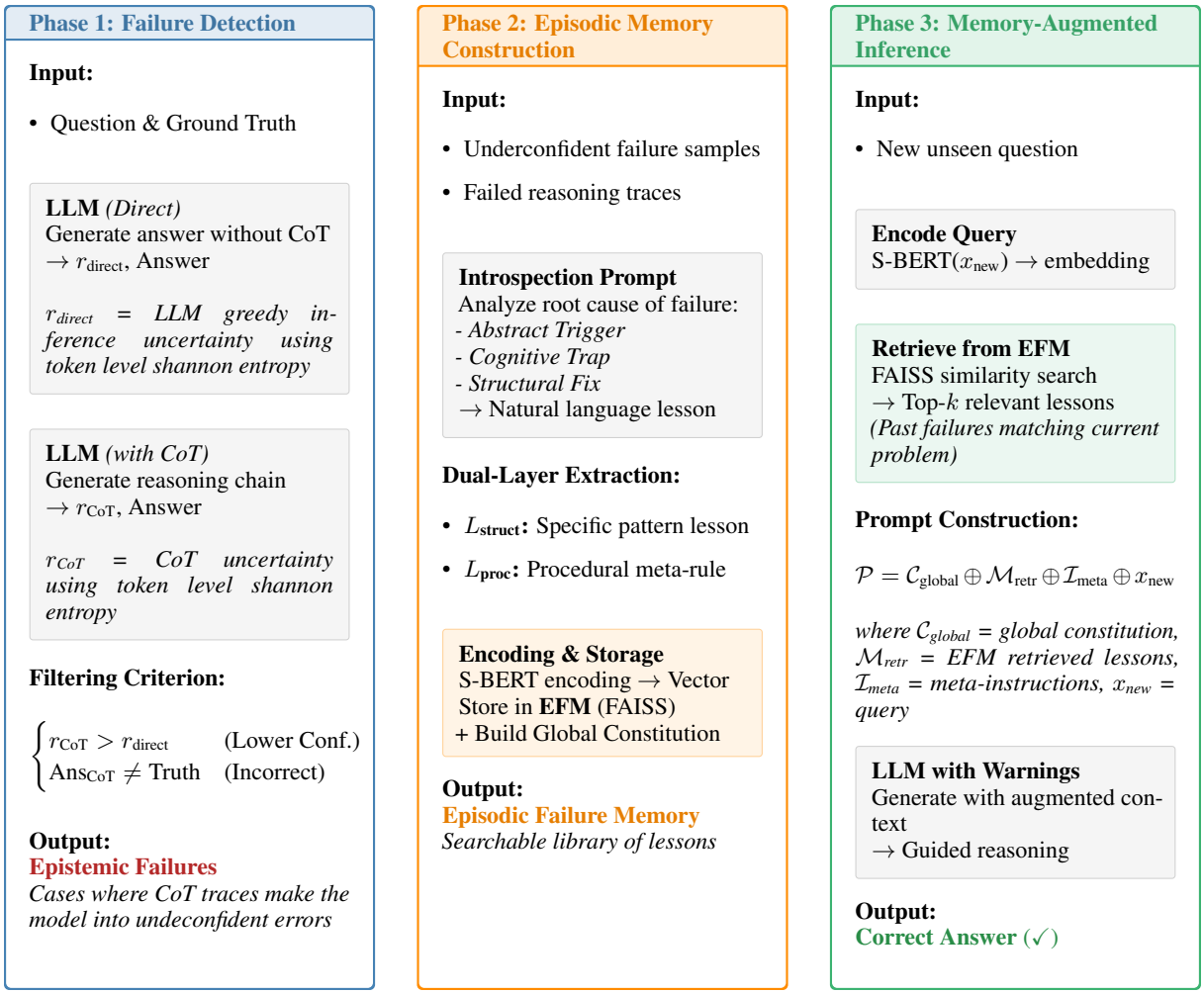


Figure 4: **The REMIND Framework.** Our pipeline operates in three phases: **(1) Failure Detection** systematically identifies epistemic failures where CoT decreases model confidence ($r_{\text{CoT}} > r_{\text{direct}}$), yielding incorrect answers; **(2) Episodic Memory Construction** triggers metacognitive introspection to extract dual-layer lessons (structural patterns and procedural rules), encoded via S-BERT and stored in a vector database (EFM); **(3) Memory-Augmented Inference** retrieves relevant past lessons for new queries, constructing a warning-augmented prompt that guides the model toward more reliable solutions.

lessons that are rarely retrieved across validation queries. This keeps the memory compact and improves retrieval specificity as the library grows.

A.1.2 L2 Constitution Pruning

The global reasoning constitution is derived by aggregating procedural failures and retaining only the most frequent rules.

A.2 Ablation Studies

This section analyzes the key design choices in REMIND and evaluates the contribution of its main components. In particular, we study: (i) the role of the dual-memory architecture, (ii) the effect of retrieval depth, (iii) the impact of retrieval strategy and embedding similarity, and (iv) the influence of memory size. These ablations isolate the factors responsible for the improvements observed in Table 1. Unless stated otherwise, all experiments follow the same setup described in Section 4.

A.2.1 Role of the Dual-Memory Architecture

REMIND separates introspective knowledge into two complementary components: an episodic memory (L1) that stores structural lessons derived from failure cases, and a consolidated knowledge store (L2) that aggregates recurring reasoning patterns into global guidance rules. To understand the contribution of each component, we compare four variants: (i) standard Chain-of-Thought reasoning without memory, (ii) L1-only retrieval using episodic lessons, (iii) L2-only reasoning using the consoli-

Symbol	Definition
x	Input question/problem
y	Ground-truth answer
$\hat{y}_{\text{direct}}$	Answer generated without CoT
$\hat{y}_{\text{CoT}}$	Answer generated with CoT
T	Number of generated tokens
$\hat{y}_{<t}$	Generated prefix before decoding step t
P_t	Next-token probability distribution at step t
$H(P_t)$	Shannon entropy at decoding step t
r	Mean token-level predictive entropy
r_{direct}	Predictive entropy under direct generation
r_{CoT}	Predictive entropy under CoT generation
Δr	$r_{\text{CoT}} - r_{\text{direct}}$
L_{struct}	Structural introspective lesson
L_{proc}	Procedural introspective lesson
D	$(L_{\text{struct}}, L_{\text{proc}})$
M	Episodic failure memory
C_{global}	Global Reasoning Constitution
x_{new}	Unseen inference-time question
$M_{\text{retrieved}}$	Top-k retrieved failure lessons
I_{meta}	Instruction block
$\oplus$	Ordered textual concatenation

Table 3: Notation used throughout the REMIND framework.

dated constitution, and (iv) the full REMIND system combining both components.

Table 4 reports the performance of these variants. This comparison reveals whether improvements arise primarily from episodic retrieval, from global reasoning rules, or from the interaction of both memory streams.

Method	Accuracy	Entropy
CoT	0.854	0.110
L1 Only	0.861	0.096
L2 Only	0.872	0.071
REMIND	0.879	0.056

Table 4: Ablation on the dual-memory architecture (GSM8K).

Global Reasoning Constitution

1. **[General Verification]**: List all given conditions and constraints explicitly before proceeding.
2. **[Step-by-Step Validation]**: Double-check each step and its validity throughout the problem-solving process.
3. **[Contextual Consistency]**: Ensure that all units, context, and domain constraints are consistent at every stage.
4. **[Pattern Confirmation]**: Verify any assumed patterns or simplifications with concrete evidence or examples.
5. **[Final Review]**: Re-evaluate all derived results against the initial conditions and constraints to ensure no oversight.

Figure 5: An example Global Reasoning Constitution (C_{global}).

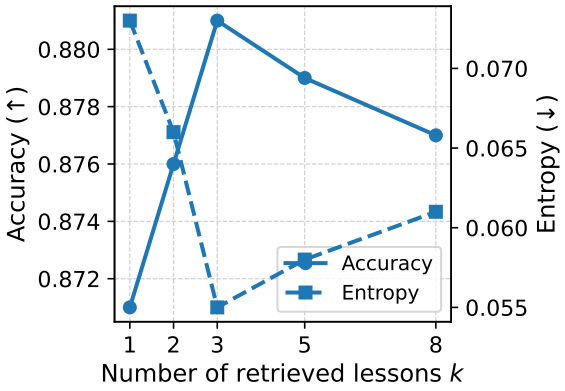


Figure 6: Accuracy and entropy as a function of retrieval depth k. Performance improves as more lessons are retrieved, then saturates or slightly declines at larger k, reflecting the trade-off between richer guidance and increased prompt complexity.

A.2.2 Effect of Retrieval Depth

During inference, REMIND retrieves the top-k most relevant lessons from the episodic memory. The number of retrieved lessons determines how much prior failure knowledge is injected into the prompt. To study the sensitivity of the system to this parameter, we vary the retrieval depth and evaluate the resulting performance.

Figure 6 shows how performance changes as the number of retrieved lessons k increases. The results illustrate the trade-off between providing richer guidance and introducing additional prompt complexity. In particular, performance typically improves as more lessons are retrieved, before saturating or slightly degrading at larger k, indicating that moderate retrieval depth provides the best balance between usefulness and distraction. We sweep $k \in \{1, 2, 3, 5, 8\}$ and use $k = 3$ as the main setting because it gives the best accuracy–entropy trade-off.

These results indicate that REMIND benefits from increasing memory size only up to a moderate regime, after which returns diminish and may slightly reverse. The likely reason is that larger memories improve coverage but also increase semantic redundancy and retrieval ambiguity, so the top-k lessons become less specific to the current query. We therefore interpret REMIND as operating in a bounded-memory regime: moderate episodic libraries are sufficient to capture common

failure patterns, while very large libraries require stronger pruning or reranking to avoid saturation.

Based on this trade-off, we use $k = 3$ in the main experiments as a stable operating point that provides useful guidance without excessive prompt clutter.

A.2.3 Retrieval Strategy and Embedding Similarity

REMIND retrieves episodic lessons using semantic similarity in embedding space. To evaluate whether the performance gains depend on this retrieval mechanism, we compare several retrieval strategies including random retrieval, lexical matching (BM25), dense semantic retrieval using sentence embeddings, and hybrid approaches that combine lexical and embedding-based signals.

Table 5 reports the performance of these retrieval methods. This experiment tests whether retrieving structurally similar reasoning lessons is necessary for effective guidance, or whether simpler retrieval strategies are sufficient. We also evaluate whether combining lexical and semantic retrieval or applying cross-encoder reranking further improves the quality of retrieved lessons.

Retrieval Strategy	Accuracy	Entropy
Random Retrieval	0.869	0.070
BM25 Retrieval	0.873	0.063
Dense Retrieval (MiniLM + Cosine/IP)	0.879	0.056

Table 5: Impact of retrieval strategy on performance.

A.2.4 Memory Size and Scaling

We further investigate how REMIND scales with the size of the episodic lesson memory. Increasing the number of stored lessons can improve coverage of diverse reasoning failures, but overly large memories may introduce retrieval noise and reduce the precision of the retrieved lessons.

Figure 7 shows the effect of memory size on reasoning performance. Performance improves as the memory grows from small to moderate sizes, indicating that broader coverage of prior failures helps retrieve more useful lessons. However, gains saturate once the memory becomes sufficiently large, and very large memories can lead to a slight degradation, suggesting a trade-off between coverage and retrieval specificity. We sweep memory sizes of 25, 50, 100, 250, and 500 lessons; accuracy peaks around 100 lessons while entropy is also lowest at that point.

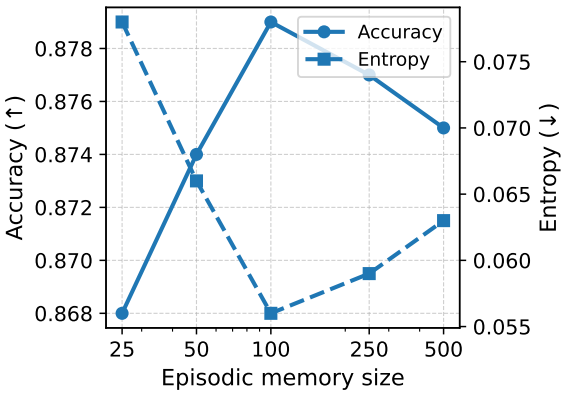


Figure 7: Effect of episodic memory size on reasoning performance. Accuracy improves and entropy decreases as the memory grows from small to moderate sizes, peaking around 100 stored lessons. Beyond this point, gains saturate and slightly reverse, suggesting a trade-off between broader failure coverage and retrieval specificity.

A.3 Symbolic Checker Experiments

To better understand the nature of reasoning errors produced by different methods, we analyze the outputs of the Chain-of-Thought (CoT) baseline and REMIND using a lightweight symbolic verification procedure. Rather than focusing only on final-answer correctness, this analysis characterizes the structure of reasoning failures and evaluates whether the reasoning trajectories produced by REMIND are more compatible with downstream verification.

A.3.1 Motivation

Incorrect reasoning outputs generally arise from different types of failures within the reasoning trajectory. In some cases, the reasoning diverges early due to incorrect intermediate deductions or logical inconsistencies. In other cases, the reasoning process is largely correct but the final answer contains a small arithmetic or transcription error.

Distinguishing between these error types is important for building reliable reasoning systems. Early reasoning collapse typically requires regenerating the entire reasoning trajectory, whereas localized execution mistakes can often be corrected automatically using symbolic verifiers or programmatic tools. Consequently, methods that produce structurally coherent reasoning trajectories—even when incorrect—are more compatible with tool-augmented reasoning pipelines.

A.3.2 Symbolic Verification Procedure

For mathematical reasoning datasets such as GSM8K, MATH, and Minerva, we construct a symbolic verification pipeline consisting of three stages:

1. **Answer Extraction.** The final numeric answer is extracted from the generated reasoning trace using a deterministic parser.

2. **Symbolic Reconstruction.** Arithmetic expressions appearing in the reasoning trajectory are evaluated using a symbolic mathematics library (e.g., `sympy`) to recover the implied numeric computations.

3. **Consistency Analysis.** The reconstructed symbolic results are compared with both the reported final answer and the ground-truth solution in order to determine whether the reasoning trajectory is internally consistent.

This procedure allows us to inspect not only whether the answer is correct, but also whether the reasoning trajectory contains structural inconsistencies.

A.3.3 Failure Type Classification

Using the symbolic checker, incorrect model outputs are categorized into three classes:

- **Early Reasoning Collapse.** The reasoning trajectory diverges from the correct solution path due to incorrect intermediate deductions or invalid logical steps.

- **Constraint Violation.** The reasoning steps contain contradictions with previously derived quantities or problem constraints.

- **Execution Slip.** The reasoning trajectory is largely correct, but the final answer contains a minor arithmetic or transcription error.

The first two categories represent structural reasoning failures, whereas execution slips correspond to localized computation errors.

A.3.4 Results

Table 6 summarizes the distribution of failure types for incorrect predictions produced by the CoT baseline and REMIND across the evaluated mathematical reasoning datasets.

REMIND substantially reduces the proportion of early reasoning collapse while increasing the share of execution-level errors. This indicates that when REMIND produces incorrect answers, the reasoning trajectory more often remains structurally consistent but contains a localized computation mistake.

A.3.5 Implications for Tool-Augmented Reasoning

This shift toward execution-level errors has important implications for reliable reasoning systems. Execution slips are typically easier to detect and correct using external verification tools, whereas structural reasoning failures require regenerating the reasoning process. Consequently, the reasoning traces produced by REMIND are more compatible with symbolic verification pipelines and programmatic correction mechanisms.

Overall, the symbolic checker analysis suggests that REMIND not only improves answer accuracy but also produces reasoning trajectories that are structurally more stable and easier to verify.

A.4 Human Evaluation Details

To assess the quality of the introspective lessons generated by REMIND, we conduct a human evaluation study. The objective is to determine whether the lessons extracted from failure cases are meaningful, generalizable, and useful for guiding future reasoning.

A.4.1 Evaluation Setup

Human annotators are presented with reasoning examples drawn from the evaluation datasets. Each example contains three components:

- the original problem statement,

- the reasoning trajectory that led to an incorrect answer,

- the introspective lesson generated by REMIND.

Annotators evaluate the lesson independently of the final answer correctness. In particular, the evaluation focuses on whether the lesson captures a meaningful reasoning insight that could help prevent similar mistakes in future reasoning tasks. Examples are presented in randomized order, and annotators are blinded to the generation method.

A.4.2 Annotation Criteria

Each lesson is evaluated along three dimensions:

Method	Early Reasoning Collapse (↓)	Constraint Violation (↓)	Execution Slip (↑)
CoT	54.7%	13.7%	31.6%
REMIND	35.2%	14.9%	49.9%

Table 6: Distribution of reasoning failure types identified using the symbolic checker.

Correctness. Whether the lesson accurately identifies the source of the reasoning failure.

Generality. Whether the lesson captures a principle that applies beyond the specific instance in which it was generated.

Usefulness. Whether the lesson would plausibly help a model or human solver avoid similar reasoning errors in future tasks.

Each dimension is scored on a 3-point Likert scale:

- 1: Poor
- 2: Moderate
- 3: High

A.4.3 Annotation Procedure

Each example is independently evaluated by multiple annotators with prior experience in mathematical or logical reasoning tasks. Annotators are provided with annotation guidelines and calibration examples before beginning the evaluation.

To reduce bias, the annotation interface hides the identity of the generation method and presents only the problem, reasoning trace, and extracted lesson. Final scores are computed by averaging the annotator ratings for each example.

A.4.4 Human Evaluation Results

Table 7 summarizes the aggregated scores across all annotated examples.

Dimension	Mean Score	Std. Dev.
Correctness	2.41	0.52
Generality	2.28	0.57
Usefulness	2.36	0.54

Table 7: Human evaluation scores for REMIND-generated introspective lessons. Scores are averaged across all annotated examples using a 3-point Likert scale.

The results indicate that the extracted lessons are frequently judged to correctly identify the underlying reasoning error while also capturing principles that generalize beyond the specific example. The usefulness scores further suggest that these lessons provide actionable guidance that could help prevent similar reasoning mistakes in future problem-solving scenarios.

A.4.5 Inter-Annotator Agreement

To assess annotation reliability, we compute inter-annotator agreement using Krippendorff's α. Agreement values range between 0.62 and 0.69 across the three evaluation dimensions, indicating substantial agreement among annotators.

A.5 Extended Experimental Details

While the experimental setup describes the data distribution, we provide the exact hyperparameters used for inference and memory retrieval to ensure reproducibility. All experiments, including the ablation studies across different model architectures, utilized a unified set of generation parameters to ensure fair comparison.

Retrieval overhead is modest relative to LLM decoding because each query requires only one sentence-embedding pass and a top-k nearest-neighbor search with $k = 3$ over 384-dimensional vectors. In our setup, the dominant cost remains autoregressive generation rather than retrieval itself. Nevertheless, as the episodic library grows further, retrieval may require additional pruning, approximate indexing, or reranking to maintain low latency.

Inference Configuration. We enabled sampling to generate diverse reasoning paths. The specific generation parameters are detailed in Table 8.

Parameter	Value
Temperature (T)	0.6
Top-p (Nucleus Sampling)	1.0 (Disabled)
Max New Tokens	4,096
Decoding Strategy	Sampling (`do_sample=True`)
Repetition Penalty	1.0 (None)

Table 8: **Generation Hyperparameters.** These settings were held constant across all baseline and REMIND experiments.

Quantization and Compute. To emulate a resource-constrained environment typical of local deployment, all models were loaded in 4-bit precision using the `bitsandbytes` library. The specific quantization configuration was:

- **Quantization Type:** NF4 (Normal Float 4)
- **Compute Dtype:** `bfloat16`
- **Double Quantization:** Enabled (to further reduce memory footprint)

Retrieval Engine Details. The *Episodic Failure Memory* relies on vector similarity search. We implemented this using the `FAISS` library with an `IndexFlatIP` (Inner Product) index, which corresponds to Cosine Similarity given normalized embeddings. The embedding model used was `sentence-transformers/all-MiniLM-L6-v2` (384 dimensions). During the inference step, the retrieval depth was set to $k = 3$ neighbors.

A.6 Compute Resources

All experiments, including memory construction, model inference, and ablation studies, were conducted on a single NVIDIA A100 GPU with 40GB of VRAM.

A.7 Datasets

We evaluate on nine reasoning benchmarks spanning arithmetic, advanced mathematics, scientific reasoning, commonsense reasoning, multi-hop question answering, and truthfulness-oriented QA. We group them into broad categories below and then briefly explain what each dataset measures and why this diversity is necessary.

Broad categories.

- **Mathematical and numerical reasoning:** GSM8K, MATH, Minerva, GPQA
- **Scientific reasoning:** ARC-C
- **Commonsense and abductive reasoning:** StrategyQA
- **Multi-hop factual reasoning:** HotpotQA
- **Truthfulness-oriented question answering:** TruthfulQA

Why multiple datasets are necessary. No single dataset is sufficient to characterize model reasoning performance, because different benchmarks stress different failure modes. A model that performs well on arithmetic problems may still struggle with scientific multiple-choice questions, multi-hop evidence aggregation, commonsense inference, or truthfulness under misleading priors. Conversely, strong performance on a knowledge-heavy or truthfulness-oriented benchmark does not imply robust symbolic reasoning or long-horizon problem solving. Using multiple datasets therefore helps distinguish whether a method improves *general reasoning behavior* or merely overfits to a narrow task format.

GSM8K. GSM8K is a benchmark of grade-school math word problems. It evaluates whether a model can carry out multi-step arithmetic reasoning, maintain intermediate quantities correctly, and reach the final numerical answer without losing consistency across steps. We include GSM8K because it provides a clean testbed for compositional numerical reasoning, where errors are easy to observe and often reflect flaws in intermediate reasoning rather than missing world knowledge alone.

MATH. MATH is a benchmark of challenging competition-style mathematics problems spanning algebra, geometry, number theory, combinatorics, and related areas. Compared with GSM8K, it requires substantially deeper symbolic manipulation and longer reasoning chains. We include MATH because it tests whether a method that helps on simpler numerical tasks continues to be useful in more demanding mathematical settings where shallow heuristics are much less effective.

Minerva. Minerva-style mathematics benchmarks focus on advanced quantitative and scientific reasoning problems that require formal manipulation, equation-based reasoning, and careful step-by-step derivation. We include Minerva because it extends the evaluation beyond standard school-level arithmetic into more technical quantitative reasoning, helping assess whether the method generalizes to harder mathematical problem distributions.

GPQA. GPQA consists of difficult graduate-level multiple-choice questions designed to be challenging even for strong language models. Compared with GSM8K, it places less emphasis on

simple arithmetic and more on expert-level conceptual reasoning and precise discrimination among closely related answer options. We include GPQA because it probes whether the method scales beyond elementary reasoning into harder, more specialized settings where superficial pattern matching is less reliable.

ARC-C. ARC-C (the Challenge split of the AI2 Reasoning Challenge) contains science exam questions that require non-trivial reasoning rather than direct lookup. Many questions combine basic scientific facts with elimination, comparison, or causal reasoning. We include ARC-C because it tests structured scientific reasoning in a multiple-choice format, complementing the free-form numerical reasoning in GSM8K and the harder expert-style questions in GPQA.

TruthfulQA. TruthfulQA evaluates whether models answer questions truthfully rather than reproducing common misconceptions or attractive falsehoods. This benchmark is important because a response can sound fluent and confident while still being factually wrong in a systematic way. We include TruthfulQA to measure whether the method improves not just reasoning fluency, but also calibration against misleading priors and popular but incorrect associations.

StrategyQA. StrategyQA contains yes/no questions that often require implicit multi-step reasoning over everyday and factual knowledge. The intermediate reasoning steps are usually not stated directly in the question, so the model must infer the latent chain needed to reach the answer. We include StrategyQA because it tests decompositional reasoning in a concise QA format, bridging the gap between explicit math reasoning and broader commonsense or factual inference.

HotpotQA. HotpotQA is a multi-hop question answering benchmark in which answering often requires combining evidence from multiple supporting facts or passages. Unlike single-hop QA, the model must connect pieces of information across sources before arriving at the final answer. We include HotpotQA because it tests compositional factual reasoning and evidence aggregation, which are central to understanding whether a method improves reasoning across linked facts rather than only within a single local context.

Overall dataset coverage. Taken together, these datasets provide a more complete picture of model capability. GSM8K, MATH, Minerva, and GPQA test numerical, symbolic, and high-difficulty reasoning; ARC-C tests scientific and exam-style reasoning; StrategyQA tests commonsense and decompositional inference; HotpotQA tests multi-hop factual reasoning; and TruthfulQA evaluates robustness against persuasive but false answers. This breadth is important because improvements on one reasoning type do not automatically transfer to others, and a strong evaluation should therefore cover multiple reasoning regimes rather than a single benchmark.

A.8 Prompts Used

The REMIND framework utilizes a pipeline of five distinct prompt templates to facilitate failure detection, metacognitive introspection, knowledge consolidation, and memory-augmented inference. Each template is designed to target a specific stage of the neuro-symbolic reasoning loop. We provide the verbatim prompts below (see Figure 10, Figure 11, Figure 12, and Figure 13). Variables enclosed in braces (e.g., `{question}`) denote slots that are dynamically filled at runtime. To ensure reproducibility, all prompts are presented in their exact format used during experimentation.

A.9 Answer-free memory and offline-label discussion.

A central design choice is that REMIND transfers *answer-free* failure knowledge rather than memorized solutions. The stored memory contains abstract triggers, cognitive traps, and verification rules; it does not store final answers or full solution templates. Memory construction is also strictly offline: ground-truth or verifier signals are used only to identify failed practice examples, while test-time REMIND requires only an embedding lookup, retrieved lessons, and one standard LLM generation. This distinction addresses two deployment concerns: the method is not a train–test memorization procedure, and it does not require labels at inference time.

A.10 Setup details on label-free construction and cross-model visibility.

Ground-truth and label-free memory construction. Ground-truth labels are used only to select failed practice examples during offline memory construction; they are never used at inference time.

Method	Full gen.	Extra LLM calls	Retrieval	Dominant online cost
CoT	1	0	No	1× decoding
Self-Consistency	40	0	No	40× decoding
Self-Refine	1	1–2	No	2–3× decoding
Meta-Prompting	1	0	No	1× decoding
RAG	1	0	Yes	1× decoding + lookup
MemPrompt	1	0	Yes	1× decoding + lookup
REMIND	1	0	Yes, $k = 3$	1× decoding + lookup

Table 9: Online inference cost comparison. REMIND uses one full autoregressive generation, matching CoT, RAG, MemPrompt, and Meta-Prompting in dominant decoding cost. Its additional cost is a lightweight pre-generation embedding and FAISS lookup, while Self-Consistency and Self-Refine require multiple online LLM generations.

This setting matches labeled development sets, historical user corrections, and automatically verifiable tasks. To test a label-free proxy, we also built memories using self-consistency pseudo-labels: the model was sampled five times on training examples and the majority-vote answer was treated as the proxy label. On GSM8K and MATH, this pseudo-label REMIND variant achieved F1 scores of 0.861 and 0.789, respectively, below the fully supervised memory and standard self-consistency. This result suggests that label-free mining is possible but currently noisy, so reliable failure signals remain important for memory construction. Although Table 1 uses Qwen2.5-7B as the main controlled backbone, cross-model robustness is evaluated explicitly in Section 4.4. We include both a difficult-subset comparison across Llama, Phi, and Qwen, and an additional Llama-3.2-8B evaluation across all eight benchmarks.

Prompt-length control discussion. The gains are also not explained by prompt length alone. Table 4 shows that both episodic structural lessons and global procedural rules contribute, with the full dual-memory design performing best. Table 5 shows that random retrieval is weaker than dense semantic retrieval; if longer context or implicit few-shot supervision were sufficient, random lessons of similar length should be competitive. Figure 6 further shows that increasing k saturates and can slightly degrade performance, which is inconsistent with a monotonic "longer prompt is always better" explanation.

A.10.1 Statistical significance of comparison between memory formation techniques

We also test the strongest comparison, Underconfident memory versus Random memory, with a paired McNemar test on GSM8K. Underconfident memory significantly outperforms Random memory ($n_{01} = 101$, $n_{10} = 52$, exact two-sided $p = 9.18 \times 10^{-5}$), where n_{01} denotes examples corrected by Underconfident memory but not by Random memory and n_{10} denotes the reverse. This provides statistical support for the epistemic-receptivity hypothesis beyond the aggregate accuracy trend.

Model Architecture	Params	CoT	REMIND	Net Δ
Small Language Models				
Llama-3.2-3B-Instruct	3B	45%	**53%**	+8%
Phi-3.5-mini-instruct	3.8B	49%	**56%**	+7%
Standard Scale				
Qwen2.5-7B-Instruct	7B	77%	**81%**	+4%

Table 10: **Architecture Generalization on a subset of MATH dataset (100 samples).** Performance of the REMIND framework across different model families and sizes. The method proves effective across the spectrum, with particularly strong error recovery in smaller models.

A.10.2 Cross model testing on Llama-3.2-8B.

To further foreground cross-model generality beyond the difficult MATH subset, we also evaluated Llama-3.2-8B-Instruct across the same eight-benchmark suite using CoT as the baseline. Table 11 shows that REMIND improves accuracy and reduces entropy in every dataset, including non-mathematical benchmarks such as ARC-C, HotpotQA, TruthfulQA, and StrategyQA.

A.11 Recoverable Errors and Tool-Readiness.

When REMIND still fails, the error is often a **terminal heuristic slip** rather than a core logic failure. In our example, the model derives the correct coordinates but misclassifies the final shape, yielding the wrong area. This distinction matters: baseline CoT fails through an incorrect method, whereas REMIND fails after mostly valid reasoning. Such localized errors are easier to detect and repair with symbolic verifiers or external tools (Section A.3), suggesting that REMIND improves not only answer accuracy but also the **tool-readiness** of the reasoning process. This qualitative picture is supported by two additional analyses. Human evaluation of the extracted lessons (Appendix A.4) gives mean scores of 2.41/3 for correctness, 2.28/3 for generality, and 2.36/3 for usefulness, with Krippendorff's α between 0.62 and 0.69, indicating that the recovered cognitive traps are generally accurate and practically useful. Complementarily, the

Dataset	Baseline CoT		REMIND	
	Accuracy ↑	Entropy ↓	Accuracy ↑	Entropy ↓
GSM8K	0.742 ±0.016	0.132 ±0.004	**0.768** ±0.015	**0.073** ±0.003
ARC-C	0.553 ±0.018	0.139 ±0.005	**0.579** ±0.017	**0.094** ±0.004
MATH	0.514 ±0.044	0.145 ±0.006	**0.542** ±0.043	**0.069** ±0.004
HotpotQA	0.160 ±0.020	0.178 ±0.008	**0.168** ±0.019	**0.134** ±0.006
GPQA	0.292 ±0.021	0.171 ±0.007	**0.330** ±0.019	**0.116** ±0.005
Minerva	0.061 ±0.013	0.197 ±0.011	**0.082** ±0.012	**0.150** ±0.008
TruthfulQA	0.520 ±0.019	0.145 ±0.005	**0.542** ±0.018	**0.104** ±0.004
StrategyQA	0.682 ±0.017	0.126 ±0.005	**0.696** ±0.016	**0.089** ±0.004

Table 11: Cross-benchmark generality on Llama-3.2-8B-Instruct. REMIND improves both accuracy and entropy over CoT across all eight benchmarks.

symbolic-checker analysis (Appendix A.3) shows that REMIND reduces early reasoning collapse (54.7% → 35.2%) and shifts more remaining errors into local execution slips (49.9% vs. 31.6%), suggesting that retrieved lessons usually keep reasoning near the correct path rather than causing new structural failures.

A.12 Extended Sensitivity to Model Architecture

To verify that the efficacy of REMIND is not an artifact of a specific model architecture, we evaluated the framework across a diverse set of LLMs with varying parameter counts. Specifically, we tested the framework on **Qwen2.5-7B** (Qwen et al., 2025), **Llama-3.2-3B-Instruct**[2], and **Phi-3.5-mini-instruct**[3].

Table 10 presents the comparative results on a difficult subset of the MATH benchmark ($N = 100$). We observe that REMIND consistently improves performance regardless of model size, though the magnitude of impact varies:

Small Language Models (SLMs). Models like Llama-3.2-3B and Phi-3.5-mini show the highest relative gains (up to +8%). These models often possess the raw reasoning capability but suffer from "contextual drift" or formatting instability. The episodic helps significantly boosting their reliability (45% → 53% for Llama).

Standard Scale Models. For Qwen2.5-7B, which already possesses a high baseline accuracy (77%), REMIND still yields a distinct improvement to **81%** (+4%). This indicates that even for capable models, the method is effective.

We additionally evaluate Llama-3.2-8B-Instruct across all eight benchmarks; REMIND improves accuracy and reduces entropy in every dataset, including non-mathematical tasks (see Table 11).

This cross-architecture consistency suggests that REMIND addresses a fundamental mechanism of context utilization in Auto-Regressive Transformers, rather than overfitting to the idiosyncrasies of a single model.

A.13 Process Fidelity and Error Recoverability

Accuracy alone hides an important qualitative shift in failure mode. REMIND changes errors from **cognitive collapse** to **structurally aligned near-misses**: even when the final answer is wrong, the reasoning is often much closer to the correct solution path (Figures 8 and 9).

Schema Hallucination vs. Structural Alignment. Baseline CoT failures often arise from *schema hallucination*, where the model adopts the wrong mathematical formulation. In geometry problems with tangency, for example, it may confuse intersection points with points of tangency, producing an invalid system from the outset. Such failures are typically unrecoverable. By contrast, REMIND more often remains **structurally aligned**: retrieved lessons help the model identify the correct constraints (e.g., tangency via point-to-line distance) and carry out the main symbolic steps correctly. See Figure 3 for reference.

Recoverable Errors and Tool-Readiness. When REMIND still fails, the error is often a **terminal heuristic slip** rather than a core logic failure. Such localized errors are easier to detect

[2] http://huggingface.co/meta-llama/Llama-3.2-3B-Instruct

[3] https://huggingface.co/microsoft/Phi-3.5-mini-instruct

and repair with symbolic verifiers or external tools (Section A.3), suggesting that REMIND improves not only answer accuracy but also the **tool-readiness** of the reasoning process. Human-evaluation and symbolic-checker details are reported fully in Appendices A.4 and A.3.

Baseline CoT Output (Failure: Cognitive Collapse)

First, we need to find the points of tangency between the circle and the parabola. The equations of the circle and parabola are: - Circle: $x^2 + y^2 = 2$ - Parabola: $y^2 = 8x$
For tangency, the system of equations must have exactly one solution for each tangent line. We substitute $y^2 = 8x$ into the circle's equation:

$$x^2 + 8x = 2 \implies x^2 + 8x - 2 = 0.$$

Using the quadratic formula $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$, where $a = 1$, $b = 8$, and $c = -2$:

$$x = \frac{-8 \pm \sqrt{64+8}}{2} = \frac{-8 \pm \sqrt{72}}{2} = \frac{-8 \pm 6\sqrt{2}}{2} = -4 \pm 3\sqrt{2}.$$

Since x must be non-negative, we take $x = -4 + 3\sqrt{2}$. Now, substituting back to find y:

$$y^2 = 8(-4 + 3\sqrt{2}) = -32 + 24\sqrt{2}.$$

Thus, the points of tangency are:

$$\left(-4 + 3\sqrt{2}, \sqrt{-32 + 24\sqrt{2}}\right) \text{ and } \left(-4 + 3\sqrt{2}, -\sqrt{-32 + 24\sqrt{2}}\right).$$

Similarly, for the other tangent, we get:

$$\left(-4 - 3\sqrt{2}, \sqrt{-32 - 24\sqrt{2}}\right) \text{ and } \left(-4 - 3\sqrt{2}, -\sqrt{-32 - 24\sqrt{2}}\right),$$

but since the latter is not valid as it doesn't lie on the circle, we only consider the first pair.

The coordinates of the points of tangency are approximately:

$$A \approx (0.343, 2.828), B \approx (0.343, -2.828), C \approx (-7.343, 2.828), D \approx (-7.343, -2.828).$$

The quadrilateral formed by these points is a rectangle with side lengths:

$$AB = 2 \times 2.828 = 5.656 \approx 5.656$$

and

$$AD = 7.343 - 0.343 = 7.$$

The area of the rectangle is:

$$5.656 \times 7 = 39.592 \approx 39.592.$$

However, the exact calculation using the exact values gives us:

$$\text{Area} = 8.$$

ANS = 8

Figure 8: The raw reasoning output from the Baseline CoT model. It incorrectly conflates the condition for tangency with curve intersection, leading to invalid coordinates and a fabricated geometric shape.

REMIND Output (Success: Structural Alignment)

To find the area of the quadrilateral formed by the points of tangency between the circle $x^2+y^2=2$ and the parabola $y^2=8x$, we need to first determine the coordinates of these points of tangency.
Step 1: Find the Points of Tangency The general equation of a tangent to the parabola $y^2=8x$ is given by $y=mx+\frac{2}{m}$. For this tangent to be also tangent to the circle $x^2+y^2=2$, it must satisfy the condition that the distance from the center of the circle (0,0) to the line $mx-y+\frac{2}{m}=0$ is equal to the radius $\sqrt{2}$.
The distance from the origin to the line $mx-y+\frac{2}{m}=0$ is:

$$\frac{\left|\frac{2}{m}\right|}{\sqrt{m^2+1}}=\sqrt{2}$$

Squaring both sides, we get:

$$\frac{4}{m^2(m^2+1)}=2 \implies \frac{2}{m^2(m^2+1)}=1 \implies 2=m^2(m^2+1) \implies m^4+m^2-2=0$$

Let $u=m^2$, then the equation becomes:

$$u^2+u-2=0 \implies (u+2)(u-1)=0 \implies u=1 \text{ or } u=-2$$

Since $u=m^2$ must be non-negative, we have $m^2=1 \implies m=\pm 1$.
For $m=1$, the tangent line is $y=x+2$. Substituting $y=x+2$ into the parabola equation $y^2=8x$, we get:

$$(x+2)^2=8x \implies x^2+4x+4=8x \implies x^2-4x+4=0 \implies (x-2)^2=0 \implies x=2$$

Thus, $y=2+2=4$. So one point of tangency is $(2,4)$.

For $m=-1$, the tangent line is $y=-x-2$. Substituting $y=-x-2$ into the parabola equation $y^2=8x$, we get:

$$(-x-2)^2=8x \implies x^2+4x+4=8x \implies x^2-4x+4=0 \implies (x-2)^2=0 \implies x=2$$

Thus, $y=-2-2=-4$. So another point of tangency is $(2,-4)$.
Similarly, for the other pair of tangents, we get the points $(-1,1)$ and $(-1,-1)$.
Step 2: Determine the Area of the Quadrilateral The points of tangency are $(2,4)$, $(2,-4)$, $(-1,1)$, and $(-1,-1)$. These points form a rectangle. The length of the rectangle is the distance between $(2,4)$ and $(2,-4)$, which is 8, and the width is the distance between $(2,4)$ and $(-1,4)$, which is 3.
The area of the rectangle is:

$$\text{Area}=8\times 3=24$$

ANS =

$$\boxed{24}$$

Figure 9: The reasoning output from the REMIND agent. It correctly applies the distance-to-center condition for common tangents and derives the exact coordinates. The final error (Area = 24 instead of 15) is a benign geometric misclassification (rectangle instead of trapezoid) rather than a fundamental logic failure.

Prompt 1: Baseline Direct Generation (No CoT)

```
Solve this problem. You can show your work, but you MUST end your response with a final line in
this exact format answering the exact question asked: ANS = <value>
Do not include any explanation or extra text on the final ANS = <value> line.
Question: {question}
```

Prompt 2: Baseline Chain-of-Thought (CoT)

```
Example:
Question: Janet has 16 apples. She gives away 4 apples. How many apples does she have left?
Let's think step by step:
Janet starts with 16 apples.
She gives away 4 apples.
So she has 16 - 4 = 12 apples left.
ANS = 12
CRITICAL INSTRUCTIONS:
1. Show your step-by-step reasoning
2. Your response MUST end with EXACTLY this format on the last line answering the exact question
asked:
ANS = <numerical_value>
3. Do NOT write anything after the ANS = line
4. Do NOT add explanations, periods, or extra text on the final line
5. The answer must be a single number after the equals sign
Now solve this problem following the exact format above:
Question: {question}
Let's think step by step:
```

Figure 10: Baseline prompts used for entropy-based failure detection. Comparison between Direct (Prompt 1) and CoT (Prompt 2) outputs identifies "reasoning illusions" where confidence increases despite incorrectness.

Prompt 3: Dual-Layer Introspection

```
You are a Senior Mathematical Logician analyzing a failure in an AI's reasoning.
[FAILURE CONTEXT]
The AI attempted this problem but failed.
Problem: {question}
AI's Reasoning: {reasoning_chain}
AI's Wrong Answer: {wrong_answer}
Correct Answer: {ground_truth}
[YOUR TASK]
Analyze this failure in TWO DISTINCT LAYERS.
=== LAYER 1: The Structural Lesson (For Vector Retrieval) ===
Extract a GENERALIZABLE mathematical lesson. Do not fixate on specific numbers (like 31, 28, f(-3)).
Instead, identify the structural pattern.
1. Concept: What mathematical concept is this? (e.g., "Binomial Symmetry").
2. Abstract Trigger: Describe the type of problem configuration that invites the error, without
using specific numbers. (e.g., instead of "x^4 and x^2", say "Polynomials containing only even
powers").
3. Cognitive Trap: Why did the CoT reasoning feel confident? What heuristic did it abuse?
4. Structure_Fix: A concise, actionable directive for future problems of this TYPE. (IF [Abstract
Condition], THEN [Specific Check]).
=== LAYER 2: The Process Flaw (For Global Constitution) ===
Identify the "Meta-Cognitive" bad habit that allowed the error to pass.
1. Process_Tag: A short 2-3 word category for the bad habit (e.g., "Constraint Blindness").
2. Process_Rule: A universal, short maxim (max 10 words) to prevent this habit in ANY math problem.
[OUTPUT FORMAT]
Concept: [Short tag]
Abstract Trigger: [Description]
Cognitive Trap: [Logic flaw]
Structure_Fix: IF [Condition], THEN [Action].
Process_Tag: [Category]


Process_Rule: [Universal Maxim]
```

Figure 11: The introspection prompt used to decompose failures into specific structural patterns (for EFM) and general procedural flaws (for the Constitution).

Prompt 4: Global Constitution Synthesis

```
You are a Mathematical Reasoning Coach.
Below is a raw list of "Bad Habits" and "Fixes" collected from various AI failure cases.
Many are repetitive (e.g., multiple mentions of "Check constraints").
[RAW LIST]
{raw_rules_text}
[YOUR TASK]
Synthesize a "Universal Reasoning Constitution" of exactly 5 RULES.
1. Merge Redundancies: "Check steps", "Verify logic", and "Validate" are the SAME rule. Combine
them.
2. Remove Domain-Specifics: Remove rules about "Geometry" unless they apply to ALL math.
3. Actionable Commands: Do not say "Be careful." Say "List constraints explicitly."
[OUTPUT FORMAT]
### UNIVERSAL REASONING CONSTITUTION
1. [Category]: [Actionable Command]
```

Figure 12: The synthesis prompt used to aggregate noisy procedural feedback into a clean, static Global Constitution.

Prompt 5: Constitution-Guided Inference

```
{constitution_text}
 # RELEVANT PAST FAILURES (Specific Patterns)
 ## Pattern 1 (similarity: {sim_score_1})
- Trap: {error_type_1}
- Trigger: {warning_signs_1}
- Fix: {lesson_1}
 ... [Additional retrieved patterns] ...
 Before answering:
 1. APPLY the Constitution rules above.
 2. CHECK against the specific failure patterns listed.
 3. If uncertain, verify constraints explicitly.
 Example:
 Question: Janet has 16 apples. She gives away 4 apples. How many apples does she have left?
 Let's think step by step:
 Janet starts with 16 apples.
 She gives away 4 apples.
 So she has 16 - 4 = 12 apples left.
 ANS = 12
 CRITICAL INSTRUCTIONS:
 1. Show your step-by-step reasoning
 2. Your response MUST end with EXACTLY this format on the last line:
 ANS = <numerical_value>
 3. Do NOT write anything after the ANS = line
 4. The answer must be a single number after the equals sign
 Now solve this problem following the exact format above:
 Question: {question}
 Let's think step by step:
```

Figure 13: The final inference prompt, which dynamically assembles the Global Constitution and retrieved failure patterns ($M_{retrieved}$) to guide the model's generation.