Learning From Past Mistakes: Contrastive Reflection Memory for Training-Free Regeneration

Anonymous authors
Paper under double-blind review

Abstract

Verification-guided self-improvement has recently emerged as a promising approach to improving the accuracy of large language model (LLM) outputs. However, existing approaches face a trade-off between inference efficiency and accuracy: iterative verification–rectification is computationally expensive and prone to being trapped in faulty reasoning, while best-of-N selection requires extensive sampling without addressing internal model flaws. We propose a training-free regeneration paradigm that leverages an offline-curated contrastive Reflection Memory (RM) to provide corrective guidance, while regenerating from scratch helps break out of faulty reasoning. At inference time, the method performs RM-guided self-verification followed by a single RM-guided regeneration, avoiding both iterative correction and multi-sample selection. We evaluated our method on nine benchmarks that span algorithmic, reasoning, symbolic, and domain-specific tasks in both small- and large-scale LLMs. Experiment results show that our method outperforms prior methods while maintaining low computational cost. Code is available at `https://anonymous.4open.science/r/Supplementary-code-5B17`.

1 Introduction

Scaling language models to billions of parameters has enabled them to encode broad domain knowledge and exhibit strong reasoning abilities. Nevertheless, ensuring the correctness of generation remains challenging. Beyond improving reasoning generation alone, recent work explores verification-guided improvement. However, without ground-truth supervision, large language models (LLMs) still struggle to reliably assess their reasoning. This gap between generation, verification, and improvement remains a major obstacle to further advancing LLMs' performance Kamoi et al. (2024); Huang et al. (2024a); Tie et al. (2025).

Extensive research has focused on improving LLMs reasoning generation. Chain-of-thought (CoT) prompting Wei et al. (2022) first demonstrated that eliciting intermediate reasoning steps can substantially improve performance. Building on this, self-consistency decoding Wang et al. (2023) further improves reasoning performance by aggregating multiple reasoning paths. Subsequently, more advanced frameworks such as Tree-of-Thought (ToT), Algorithm-of-Thought (AoT), and Forest-of-Thought (FoT) Yao et al. (2023); Sel et al. (2024); Bi et al. (2025) were developed, which improve reasoning capabilities by encouraging explicit search over intermediate steps. Additionally, numerous studies have shown that reasoning performance can be further improved through well-designed few-shot demonstrations. Automatic demonstration generation Zhang et al. (2023) automates the construction of reasoning demonstrations, while contrastive ICL approaches Chia et al. (2023); Mo et al. (2024); Gao & Das (2024) leverage paired correct and incorrect reasoning examples to highlight common reasoning failures. However, these methods primarily focus on reasoning generation, leaving verification and rectification underexplored.

A complementary line of research targets reflection and refinement to further improve LLM performance, as summarized in Table 1. Training-free approaches such as Self-Refine Madaan et al. (2023) rely on LLM to self-verify, while Reflexion Shinn et al. (2023) uses ground-truth labels for verification. These methods iteratively refine output using LLM-generated feedback. CRITIC Gou et al. (2024) and Self-Debugging Chen et al. (2024) focus on specific tasks (like algorithmic reasoning, code debugging, or knowledge-based QA),

Method	Learning	Verification	Improvement Method	Iterate
Training-Free				
Self-Refine Madaan et al. (2023)	ICL	self-critic	Explicit FB Rectify	✓
Reflexion Shinn et al. (2023)	ICL	Oracle	Explicit FB Rectify	✓
ProCo Wu et al. (2024)	ICL	Solve Verif. Q.	Implicit Rectify	✓
ST CoT Wu et al. (2025)	ICL	Uncertainty-guided	Implicit Rectify	✓
SETS Chen et al. (2025)	ICL	Self-verify	Explicit FB Rectify+MV	✓
Once-More Gao et al. (2026)	ICL	Perplexity-guided	Implicit Rectify	✓
TF-TTCL Zheng et al. (2026)	ICL	Perplexity-guided	Online Rule-guided Gen.	✗
Ours	ICL	RM-guided	RM-guided Regenerate	✗
External Tool				
CRITIC Gou et al. (2024)	ICL	Tool-Assist	Explicit FB Rectify	✓
Self-Debugging Chen et al. (2024)	ICL	Programmatic	Explicit FB Rectify	✓
Training-Enabled				
V-STaR Hosseini et al. (2024)	Fine-tuning	Trained Verifier	Best-of-N	✓
ReflectEvo Li et al. (2025)	Fine-tuning	Oracle	Implicit Rectify	✗
SCoRe (RL) Kumar et al. (2025)	RL fine-tuning	✗	Implicit Rectify	✗

Table 1: Comparison across major verification-guided improvement methods. ✓/✗ indicate presence/absence of a component; "Oracle" denotes access to ground-truth labels; "Rectify" vs. "Regenerate" indicate whether prior outputs are reused; "Explicit" vs. "Implicit" indicate whether textual feedback is produced. FB and MV stand for feedback-based and majority voting, respectively.

using external tools such as a search engine, a calculator, or programmatic execution to verify, then rectifying iteratively with LLM-generated feedback. Self-Train CoT (ST CoT) Wu et al. (2025) instead uses entropy-based uncertainty for verification and implicitly refines responses by prompting the LLM to "think deeper" through adaptive iterations. Recent work has further explored scaling and test-time adaptation. SETS Chen et al. (2025) scales iterative self-verification and correction by sampling multiple candidate responses and applying majority voting to aggregate them. TF-TTCL Zheng et al. (2026) employs multi-role test-time exploration to distill contrastive rules online, which accumulate across the test stream and guide subsequent generation. In addition, fine-tuning–based methods have also been explored. V-STaR Hosseini et al. (2024) trains a verifier to select the best response from multiple candidates. ReflectEvo Li et al. (2025) improves reasoning through supervised reflection learning. SCoRe Kumar et al. (2025) further investigates reinforcement learning for self-correction.

Despite recent progress, existing approaches still exhibit several limitations in achieving practical, reliable, and efficient verification-guided improvement. First, many self-refinement methods, such as ReflectEvo, Reflexion, CRITIC, and Self-Debugging rely on ground-truth, trained verifiers, or external tools for verification, limiting their applicability for general inference. Second, best-of-N selection methods, like V-STaR Hosseini et al. (2024) leave internal flaws unaddressed when all responses have errors. Although iterative approaches could mitigate this issue but introduce substantial computational overhead. Third, iterative refinement approaches, such as Self-Refine, ST CoT perform unsupervised multi-step refinement, accumulating errors in feedback and rectification while incurring high computational cost. These limitations motivate a rethinking of verification-guided improvement: how to enable reliable self-verification and refinement without ground-truth supervision, while avoiding costly iterative generation thereby reducing inference cost and improve real-world applicability.

To address these challenges, we introduce Contrastive Reflection Memory (RM), which organizes corrective insights from a reflection model into a structured repository of base-model errors (Sec.2.1). During inference, relevant RM entries are retrieved (Sec.2.2.1) and incorporated into the reasoning process in a training-free manner: (i) We first study a simple RM-primed prompting strategy to verify that retrieved reflection memories contain useful reasoning signals (Sec.2.2.2); and (ii) Building on this observation, we propose RM-Regen, a verification-guided regeneration framework that selectively triggers RM-guided regeneration only when needed. (Sec.2.2.3). Extensive experiments across multiple LLMs demonstrate the applicability of our method. Without fine-tuning, external tools, or ground-truth label, our method achieves the best performance with lower computational cost (Sec. 3). In summary, our main contributions are:

- **Contrastive Reflection Memory Curation.** We introduce offline-curated RM that transforms corrective insights from a reflection model into reusable principles and contrastive examples. This enables base models to leverage reflection-level heuristics for inference-time self-improvement.
- **Enhanced Self-verification.** We introduce a verifier guided by retrieved RM and model uncertainty to provide reliable verification signals without requiring ground-truth labels or external tools.
- **Efficient Single-step Regeneration.** We propose an RM-guided regeneration paradigm that replaces iterative editing with a single regeneration step, improving reasoning reliability while achieving a better cost–performance trade-off than sampling-heavy or multi-refinement methods.

2 Training-Free Regeneration Paradigm

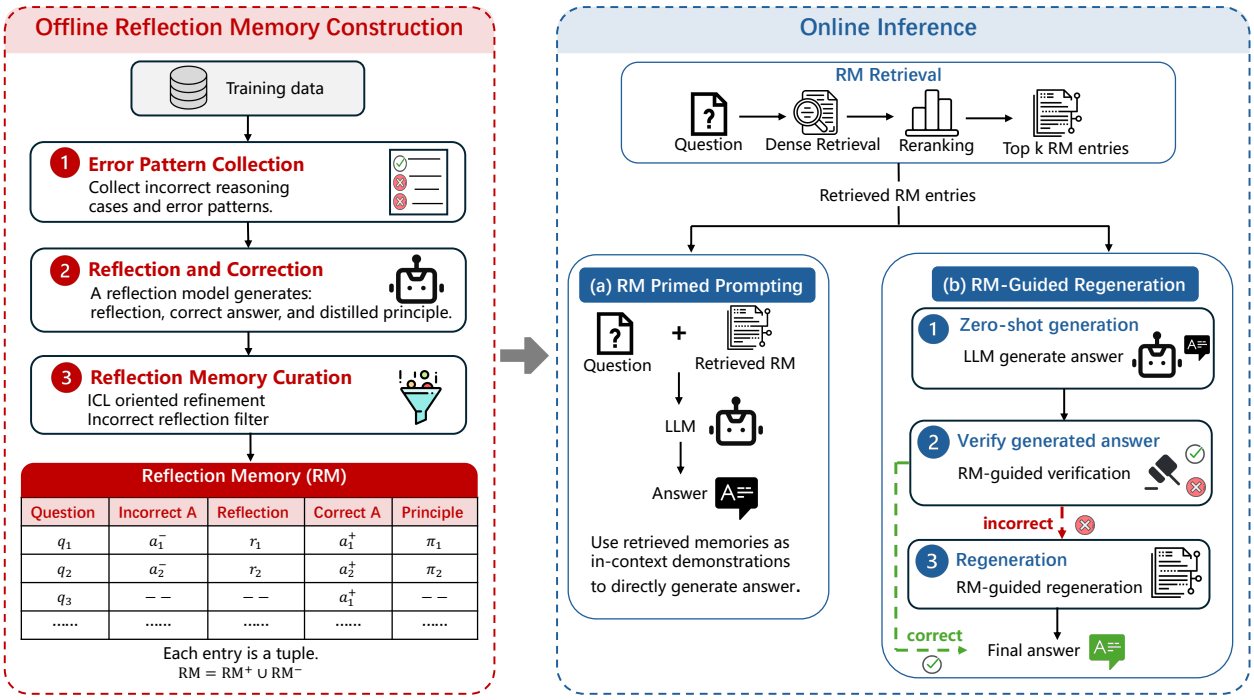


Figure 1: Overview of our training-free regeneration framework. Left: Offline Reflection Curation. Right: Online Inference. Based on retrieved RM entries: (a) Direct Performance Boost: the LLM answers the query conditioned on the retrieved memory context. (b) RM-Guided Regeneration: the LLM first generates and verifies an answer; if incorrect, we retrieve reflections and regenerate a corrected answer conditioned on the memory context.

As illustrated in Figure 1, our method operates in two stages. **(1) Offline Reflection Memory Curation.** First, we collect erroneous reasoning patterns from training data. A reflection LLM then reflects on and rectifies these errors. The resulting reflections and rectifications are subsequently refined and filtered to construct a structured contrastive RM. **(2) Online Inference.** At inference time, relevant RM are retrieved based on embedding similarity and can be used in two variants: (i) RM-Primed, RM-guided pre-reasoning primed prompting or (ii) RM-Regen, RM-guided post-reasoning verification and regeneration.

2.1 Offline Reflection Memory Curation

This section details the curation of contrastive RM from base-model errors, reflections, and an ICL-oriented refinement and filtering pipeline.

2.1.1 Error Pattern Collection

Let Q denote the questions in the training set. For each $q \in Q$, we run the base LLM for k times and obtain responses: $M_{base}(q) = \{a_1, a_2, \ldots, a_k\}$. When all responses match the ground truth, we label the pair (q, a_1) as a correct demonstration (we take the first instance without loss of generality). If responses are inconsistent, it indicates the model has the required knowledge but exhibits stochastic reasoning variance. In such cases, we retain the response a_i that matches the ground truth and label (q, a_i) as a clean supervision signal, filtering out incorrect attempts to provide an unambiguous reference that reduces uncertainty and reinforces reliable reasoning paths. When all responses are incorrect, we label (q, a_1) as an error-prone case, reflecting potential deficiencies in the model's reasoning or knowledge.

2.1.2 Reflection Generation

A LLM is employed as a reflection model to analyze erroneous reasoning a^-, reflect underlying mistakes r, and provide corrected reasoning traces a^+: $(r, a^+) = M_{\text{reflection}}\big(q, a^-; \tau_{\text{reflect}}\big)$. In addition, the reflection model is prompted to derive concise and generalizable principles π from the reflections, which can help improve future reasoning performance: $\pi = M_{\text{reflection}}\big(r; \tau_{\text{principle}}\big)$. The prompt template $\tau_{\text{principle}}$, τ_{reflect} used for reflection are provided in Appendix B.2 and Appendix B.1.

2.1.3 Filtering and Refinement

Directly using the collected pairs of incorrect reasoning, reflection, and rectification is suboptimal. Incorrect reasoning is often verbose, inconsistent with the rectified reasoning format, and may contain compounded errors, which reduces the effectiveness of ICL. Moreover, reflections and rectifications can be unreliable. To address these issues, we introduce a filtering step to remove cases with incorrect rectifications, and an ICL-oriented refinement step to clean, normalize, and structure these examples for in-context use.

ICL-Oriented Refine Motivated by rewrite-then-regenerate Shao et al. (2023) and reverse-engineered reasoning Wang et al. (2025), we propose ICL-oriented refinement, where the reflection LLM rewrites each incorrect response a^- into a compact contrastive negative $\hat{a}^-$ by removing verbosity, preserving the incorrect final answer, and explicitly surfacing the specific error identified in reflection: $\hat{a}^- = M_{\text{reflection}}(a^-, r, a^+; \tau_{\text{refine}})$. The refined prompt template τ_{refine} is provided in Appendix B.4. This refinement concentrates the error signal and sharpens the contrast between incorrect and corrected reasoning, improving contrastive ICL effectiveness. To conceptually illustrate the intuition, we define the "information contribution" of a refined negative and its reflection as:

$$\mathcal{I}(\hat{a}^-; r, a^+) \approx p_{M_{\text{base}}}\big(y^* \mid q', \text{ICL}(q, \hat{a}^-, r, a^+)\big). \tag{1}$$

where $\mathcal{I}(\hat{a}^-; r, a^+)$ denotes the base model's prob- ability of predicting the correct answer y^* for a test question q' when prompted with the contrastive demonstration.

2.1.4 Reflection Memory Structuring

Finally, we organize the curated contrastive RM into two subsets, correct entries $\mathcal{R}^+$ (validated solutions) and incorrect entries $\mathcal{R}^-$ (reflections and rectifications), as follows:

$$\mathcal{RM} = \mathcal{R}^+ \cup \mathcal{R}^-, \tag{2}$$

$$\mathcal{R}^+ = \{\, (q, a^+, v) \mid v = \text{correct} \,\}, \tag{3}$$

$$\mathcal{R}^- = \{\, (q, \hat{a}^-, v, r, a^+, \pi) \mid v = \text{incorrect} \,\}. \tag{4}$$

Here q denotes question; $\hat{a}^-$ is refined incorrect answer; v is verification; r is reflection; a^+ is corrected answer; and π is the principle summarized from reflection. Unlike conventional contrastive ICL, which stores only positive and negative demonstrations, each RM entry explicitly records the verification outcome, failure analysis, corrective reasoning, and distilled principles. As a result, RM serves not only as an example repository for in-context learning, but also as a reusable source of corrective knowledge that supports both RM-guided verification and RM-guided regeneration.

2.2 Online Inference

In this section, we describe how reflection memories are retrieved and incorporated during inference. We first present an RM-primed prompting strategy as a simple way to leverage retrieved memories. We then extend this to RM-guided verification and regeneration, which introduces a selective improvement mechanism.

2.2.1 Reflection Memory Retrieval

At inference time, we use a retrieval-then-reranking pipeline to select the most relevant RM. First, we use Contriever Izacard et al. (2021), a pretrained contrastive-learned dense retriever, to retrieve a coarse top-n candidate set by embedding similarity between the test query q' and a^+ in RM You et al. (2024). We then use MPNet-based sentence encoder Song et al. (2020) to rerank these candidates by semantic similarity between q' and q in RM, and retain the top-k entries as the final memory context. This combines Contriever for broad, reasoning-oriented retrieval with MPNet for fine-grained question matching.

Following Huang et al. (2024b), our RM retrieval-augmented inference can be interpreted through a reinforcement learning (RL) perspective without explicitly performing RL training: a model's output distribution under a given context resembles a context-dependent policy. Specifically, the task context q' plays the role of state S, and the generated answer a' corresponds to action A. By retrieving the relevant RM demonstrations, the model conditions on previous validation solutions and rectification strategies via ICL, thereby guiding the conditional distribution toward more effective outputs. Under this interpretation, the implicit policy induced by memory-guided inference can be approximated as:

$$\pi(A = a' \mid S = q') \;\approx\; P\big(a' \mid q',\, \mathcal{R}^{+}_{\text{top}} \cup \mathcal{R}^{-}_{\text{top}}\big), \tag{5}$$

where $\mathcal{R}^{+}_{\text{top}}$ and $\mathcal{R}^{-}_{\text{top}}$ denotes retrieved top-k memory contexts from $\mathcal{R}^{+}$ and $\mathcal{R}^{-}$.

2.2.2 RM-Primed Prompting

In RM-Primed Prompting, the LLM implicitly performs reasoning, verification, and rectification in a single forward pass, guided by retrieved RM context. Without memory guidance, the model directly generates an answer based solely on the query q'. With RM guidance, generation is additionally conditioned on retrieved contrastive memories:

$$P\big(a' = y^* \mid q',\, \mathcal{R}^{+}_{\text{top}} \cup \mathcal{R}^{-}_{\text{top}}\big), \tag{6}$$

where y^* denotes ground truth answer. This indicates that RM shifts the conditional distribution of the model, providing actionable signals that boost performance without additional fine-tuning.

2.2.3 RM-Guided Verification and Regeneration

Since some tasks can be solved correctly without additional context, injecting RM may be redundant and can even disrupt an initially correct reasoning process. In the RM-Guided Regen variant, we explicitly separate verification and regeneration, then trigger regeneration only for responses verified as incorrect.

RM–Guided Verification Unlike approaches that rely on ground truth, stronger external judges, or model uncertainty alone, our RM-guided self-verification grounds error detection in retrieved reflection memories, enabling the model to assess its reasoning against previously observed failures and their corrections. Guided by retrieved RM, the same base model first compares its answer with relevant RM entries to detect errors in high precision: $v_{init} = M_{\text{base}}(q', a', \mathcal{R}^{+}_{\text{top}} \cup \mathcal{R}^{-}_{\text{top}}; \tau_{\text{verify}})$. where $v_{\text{init}} \in \{\text{correct}, \text{incorrect}\}$. Then an entropy-based uncertainty check is applied to recover recall by flagging doubtful "correct" answers in v_{init} and derive a final verification:

$$v = \begin{cases} \text{incorrect}, & v_{\text{init}} = \text{incorrect} \\ \text{incorrect}, & v_{\text{init}} = \text{correct} \wedge H(q', a') > \gamma \\ \text{correct}, & \text{otherwise.} \end{cases}$$

This combination of memory-based filtering and uncertainty-based recovery yields a favourable precision–recall trade-off without fine-tuning, ground truth, or an external judge. Retrieved RM therefore serves a dual role: it improves error detection during verification and provides corrective guidance for subsequent regeneration.

RM–Guided Regeneration Responses verified as incorrect are then regenerated from scratch, conditioned on the retrieved RM as: $a'' = M_{\text{base}}(q', \mathcal{R}^{+}_{\text{top}} \cup \mathcal{R}^{-}_{\text{top}})$. Unlike RM-Primed, regeneration is invoked only after explicitly verified as incorrect, preventing unnecessary interference with correct reasoning while focusing corrective computation on genuinely erroneous cases.

3 Experiments

3.1 Experiment Setup

We evaluate our method on diverse benchmarks and models, focusing on overall accuracy, verification quality, and efficiency.

Datasets (i) *Algorithmic*: GSM8K Cobbe et al. (2021), GSM-Hard Shu et al. (2024), and MATH (Level 3 subset) Hendrycks et al. (2021); (ii) *Common-sense reasoning*: StrategyQA Geva et al. (2021) and Bamboogle Press et al. (2023); (iii) *Symbolic reasoning*: Coin Flip and Letter tasks Wei et al. (2022); and (iv) *Domain-specific tasks*: LegalBench (LB) Guha et al. (2023) and News Headline (HL) Sinha & Khandait (2021).

Baselines For RM–Primed, which performs a single forward pass, we compare to Contrastive CoT (paired positive–negative demonstrations). For RM–Regen, which involves generation, verification, and regeneration, we compare with: Reflexion (reflection with episodic memory), ReflectEvo (fine-tuning on reflection-augmented data), Best-of-N (selection among candidates), Self-Refine (iterative self-assessment and refinement), ProCo (verification-question solving and error-guided rectification), and ST-CoT (uncertainty-guided deeper CoT), SETS (self-verification with multi-sample majority voting), TF-TTCL (test-time contrastive rule distillation and retrieval).

Models We conduct experiments with three base models: a large-scale model, GPT-3.5, and two smaller open-weight models, Llama-3.1-8B, Gemma-2-9B, which represent diverse architectural families; a reflection model: GPT-5mini. Additional implementation details, including dataset splits, inference and hyperparameter settings, are provided in the Appendix A.

3.2 RM-Primed Prompting

	Algorithmic			**Reasoning**		**Symbolic**		**Domain**		**Overall**
Method	GSM8K	GSM-H	MATH	SQA	Bamb	Coin	Letter	LB	HL	avg%
GPT-3.5										
Few-shot CoT	79.25	41.25	51.60	61.50	55.00	74.25	72.00	44.21	70.00	61.15
Contrastive CoT (w/o r)	77.20	39.40	47.00	60.75	49.00	59.00	75.33	35.75	72.00	57.01
Contrastive CoT (w r)	78.40	38.00	52.20	63.00	50.00	69.25	73.33	55.79	71.00	60.33
RM-Primed ($\mathcal{R}^{+}$)	81.50	42.50	**55.00**	65.25	59.00	78.00	**84.00**	64.21	70.00	64.99
RM-Primed	**81.80**	**44.60**	54.80	**68.75**	**62.00**	**79.75**	76.00	**67.37**	**72.00**	**66.01**
Llama 3.1-8B										
Few-shot CoT	84.80	31.80	**71.40**	69.25	51.00	65.75	68.00	48.42	67.00	64.19
Contrastive CoT (w/o r)	76.2	33.20	66.00	69.25	25.00	60.75	61.33	57.89	66.00	59.56
Contrastive CoT (w r)	76.40	33.00	67.60	70.25	29.00	66.25	62.67	53.68	73.00	61.13
RM-Primed ($\mathcal{R}^{+}$)	84.60	**35.80**	64.40	70.25	51.00	78.75	72.00	62.11	72.00	65.94
RM-Primed	**85.20**	34.20	70.00	**73.25**	**54.00**	**80.75**	**81.33**	**64.21**	**73.00**	**68.23**

Table 2: Accuracy of different direct-boost methods on representative benchmarks and overall average accuracy. RM-Primed ($\mathcal{R}^{+}$) denotes that only correct entries are used. w/ r and w/o r indicate with and without reflection.

We first evaluate RM-Primed against baselines without explicit verification or rectification, as shown in Table 2. For GPT-3.5, our approach obtains the highest average accuracy of 66.01%, yielding a 4.86% point improvement over Few-shot CoT. For LLaMA 3.1-8B, our method reaches an average accuracy of 68.23%, a 4.04% gain compared to Few-shot CoT. Moreover, contrastive RM is more effective than RM with only correct entries. Compared with contrastive CoT, we observe that: (i) simply adding fixed contrastive examples is ineffective; retrieving the most relevant contrastive examples is crucial for consistent gains across benchmarks, and (ii) contrastive examples augmented with reflections are more beneficial than those without. Overall, these results indicate that our RM-primed prompting provides a stronger direct boost than both standard CoT and contrastive CoT baselines.

3.3 RM-Guided Verification and Regeneration

We further compare RM-Regen with several verification-based baselines. In the oracle setting, we include Reflexion, ReflectEvo, and Best-of-N selection (upper bound), while in the non-oracle setting, we evaluate against Self-Refine, ProCo, ST-CoT, SETS, and TF-TTCL.

3.3.1 With Oracle Verification

Under the oracle verification setting, as shown in Table 3, our method achieves the best overall performance. For GPT-3.5, RM-Regen obtains the highest average accuracy of 76.94%, outperforming Reflexion (3 iterations) by 6.29%, and Best-of-N (N=3) by 9.65%. A similar trend holds for LLaMA 3.1-8B and Gemma 2-9B. RM-Regen achieves 75.85% on LLaMA 3.1-8B, surpassing Reflexion (3 iterations) by 2.59%, Best-of-N (N=3) by 6.77%, and ReflectEvo by 11.11%. On Gemma-2-9B, RM-Regen reaches 71.42%, improving over Reflexion (3 iterations) by 4.02% and Best-of-N (N=3) by 11.64%. These gains indicate that our training-free regeneration can be more effective than iterative refinement and multi-generation strategies, and more generalizable than training-based approaches.

	Algorithmic			**Reasoning**		**Symbolic**		**Domain**		**Overall**
Method	GSM8K	GSM-H	MATH	SQA	Bamb	Coin	Letter	LB	HL	avg%
GPT-3.5										
Reflexion(3 iters)	85.60	45.20	62.25	71.00	66.00	**88.25**	78.67	**85.26**	72.00	70.65
Best-of-N (N=3)	84.60	46.80	**64.00**	71.50	63.00	77.00	74.67	32.63	70.00	67.29
RM-Regen	**90.20**	**68.60**	63.40	**77.25**	**73.00**	85.75	**87.33**	75.79	**73.00**	**76.94**
Llama 3.1-8B										
Reflexion(3 iters)	89.60	36.20	77.80	**85.00**	60.00	81.25	80.67	**81.05**	70.00	73.26
ReflectEvo	85.40	32.00	73.60	71.25	53.00	67.75	68.67	50.53	62.00	64.74
Best-of-N (N=3)	89.60	34.20	76.80	77.50	53.00	72.00	70.67	64.21	75.00	69.08
RM-Regen	**92.60**	**41.00**	**80.60**	83.25	**63.00**	**85.75**	**89.33**	66.32	**75.00**	**75.85**
Gemma2-9B										
Reflexion(3 iters)	91.20	50.80	72.00	67.50	50.00	60.00	**51.33**	75.79	71.00	67.40
ReflectEvo	85.80	44.40	63.80	65.00	47.00	67.25	42.14	48.42	72.00	62.92
Best-of-N (N=3)	88.80	48.40	67.20	53.75	46.00	49.00	34.67	45.26	67.00	59.78
RM-Regen	**92.00**	**53.00**	**72.20**	**73.50**	**53.00**	**78.25**	44.29	**78.95**	**73.00**	**71.42**

Table 3: Accuracy on nine benchmarks under oracle verification, comparing RM-Regen with verification-and-rectification and selection-based baselines.

Figure 2 presents additional results for Reflexion with 2–5 iterations and Best-of-N sampling ($N = 3$ and $N = 10$). We observe that under oracle verification, the performance of Reflexion generally improves as the number of refinement iterations increases. However, Reflexion only approaches the performance of our method after five iterations, whereas RM-Regen requires only a single regeneration step without iterative refinement. For the selection-based baseline, Best-of-N achieves higher accuracy as the number of sampled candidates increases. Nevertheless, even with $N = 10$, its performance still remains below that of RM-Regen. Overall, these results indicate that given the same perfect verification signal, our method effectively transforms it into larger gains.

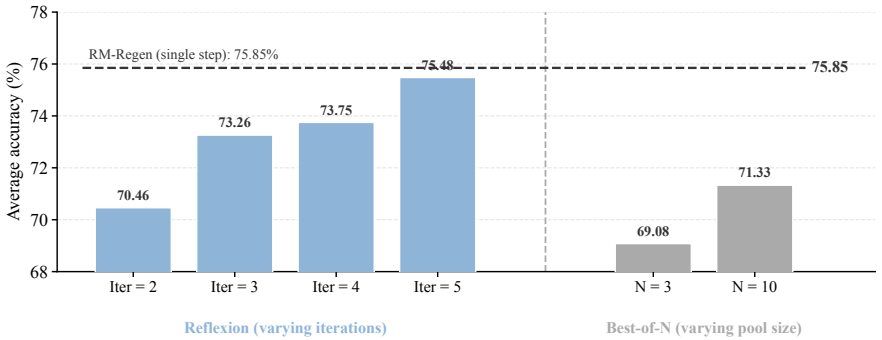
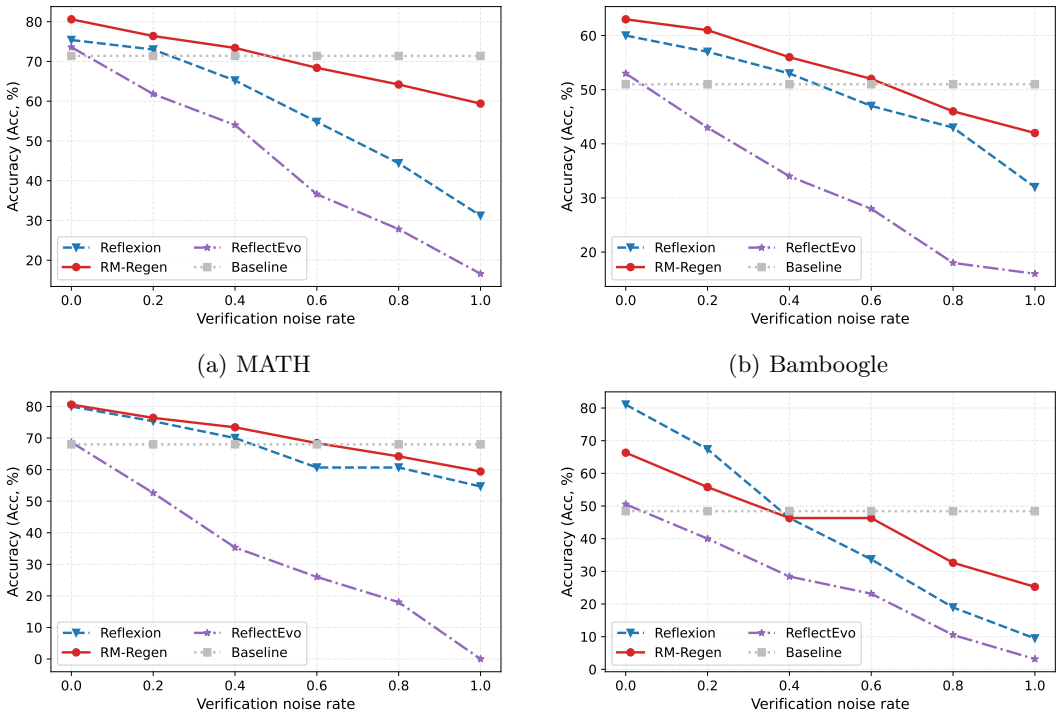


Figure 2: Average accuracy under oracle verification on Llama 3.1-8B. Reflexion improves with more refinement iterations, while Best-of-N benefits from larger candidate pools. RM-Regen achieves stronger performance with a single regeneration step.



(a) MATH (b) Bamboogle

(c) Letter (d) Legal Bench

Figure 3: Performance robustness against verification noise. We compare our method with Reflexion and ReflectEvo on Llama 3.1-8B. The gray baseline denotes Few-shot CoT. The x-axis represents the noise rate, where 0 denotes oracle verification and 1 denotes verification is always incorrect.

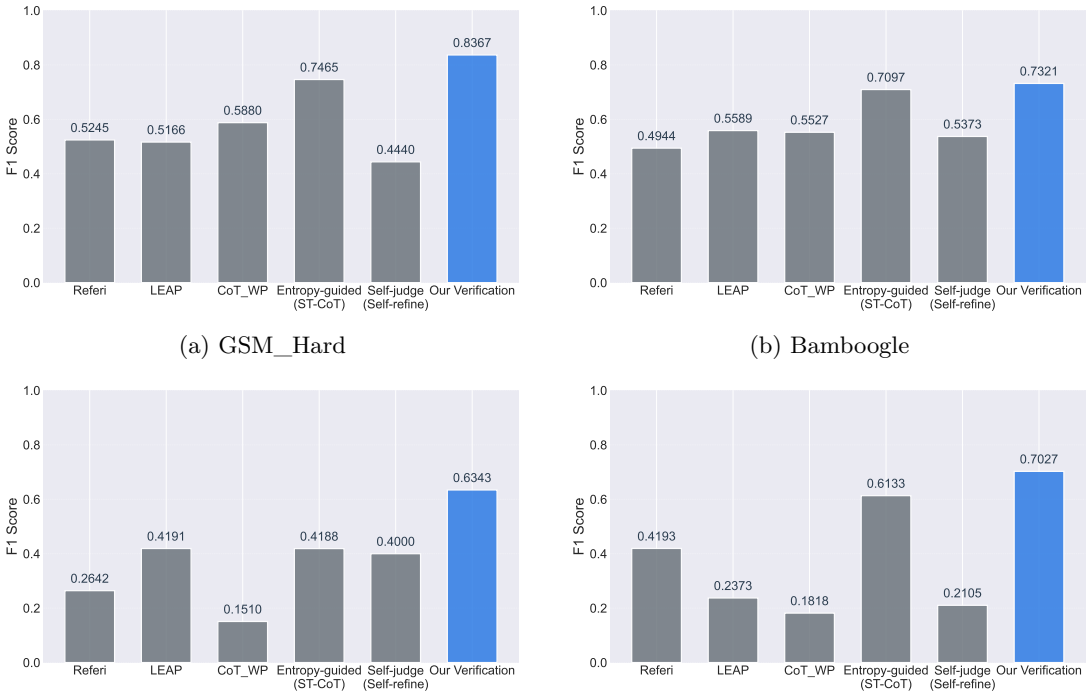
Since oracle verification is impractical in real-world settings, we analyze the impact of imperfect verification by progressively injecting noise into the oracle verifier. Verification noise is simulated by independently flipping each oracle label with a probability of the noise rate, as illustrated in Figure 3. It shows how different methods behave under varying levels of verification noise. As expected, the accuracy of all methods decreases monotonically as the verification noise rate increases. However, the rate of degradation differs substantially across methods. At zero noise, our method already achieves the highest accuracy, outperforming Reflexion and ReflectEvo. As the noise rate grows, the performance of both Reflexion and ReflectEvo drops sharply. In contrast, our method exhibits a much flatter decay curve and maintains a clear performance margin across all noise rates. Even when the verification signal becomes highly unreliable, our approach still preserves a

substantial portion of its original accuracy. These trends indicate that our contrastive reflection mechanism is significantly more robust to incorrect verification signals than prior methods.

3.3.2 Without Oracle Verification

Under no oracle verification setting, we first evaluate our verification performance against existing verifiers: Referi Lee et al. (2025), LEAP Zhang et al. (2024), CoT_WP Wang & Zhou (2024), Entropy-guided (ST CoT), Self-judge (Self-Refine). Then, we evaluate the overall accuracy of verification-guided improvement methods, including Self-Refine, ProCo, ST-CoT, and SETS, as well as TF-TTCL, which improves generation through test-time contrastive rule distillation and retrieval.

Verifier Comparison Figure 4 compares the verification F1 scores of different verification strategies on GSM_Hard, Bamboogle, Coin, and Legal. Our verifier attains an F1 score outperforming the strongest baseline, the entropy-guided verifier from ST-CoT. Other methods such as Referi, LEAP, CoT_WP, and the self-judge verifier lag considerably behind. The consistent gains across algorithmic, reasoning, symbolic and specific domain benchmarks suggest that our verifier generalizes well across tasks. Overall, these results demonstrate that our verification scheme provides a more reliable signal for subsequent rectification.



(a) GSM_Hard (b) Bamboogle

(c) Coin (d) Legal Bench

Figure 4: Verification F1 score of different verifiers on representative benchmarks using Llama-3.1-8B.

Overall Accuracy Comparison Table 4 reports the performance of different verification-and-rectification methods without oracle verification signals. For GPT-3.5, our method achieves the highest overall average accuracy of 69.87%, which represents a substantial 8.85% improvement over the best baseline, ST CoT (3 iterations, 61.02%). A similar pattern is observed for LLaMA 3.1-8B and Gemma 2-9B. Our approach attains an average accuracy of 68.05% on Llama 3.1-8B, outperforming TF-TTCL (N=3, 64.26%) by 3.79%, and 64.84% on Gemma 2-9B, exceeding the strongest ProCo variant (3 iterations, 60.87%) by 3.97%. These results indicate that, under imperfect verification, directly regenerating answers with retrieved RM is more robust than reflection-style iterative "explain-and-fix". The effectiveness of reflection-style methods depends heavily on the quality of intermediate verification and feedback, which may become unreliable and further propagate errors. Similarly, increasing the number of sampled candidates alone does not consistently improve performance in the absence of reliable verification and corrective guidance. Moreover, the retrieved RM

	Algorithmic			Reasoning		Symbolic		Domain		Overall
Method	GSM8K	GSM-H	MATH	SQA	Bamb	Coin	Letter	LB	HL	avg%
GPT-3.5										
Self-Refine (3 iters)	70.40	32.80	47.20	41.75	42.00	68.25	46.00	28.42	53.00	50.38
ProCo (3 iters)	80.80	39.60	52.20	60.00	43.00	75.25	74.00	41.05	65.00	60.55
ST CoT (3 iters)	80.20	39.80	53.60	62.00	59.00	75.00	72.67	23.16	69.00	61.02
SETS (3 iters, N=3)	83.80	39.20	53.60	62.25	56.00	63.75	72.14	25.26	69.00	59.90
TF-TTCL (N=3)	80.00	37.40	**57.60**	47.25	51.00	70.75	66.43	47.37	**72.00**	58.82
RM-Regen	**83.60**	**64.00**	56.80	**69.75**	**59.00**	**78.50**	**77.33**	**60.00**	71.00	**69.87**
Llama 3.1-8B										
Self-Refine (3 iters)	75.60	23.00	51.00	53.75	49.00	66.00	40.00	46.32	65.00	52.64
ProCo (3 iters)	83.40	31.20	69.60	62.00	39.00	44.00	71.33	38.95	48.00	57.41
ST CoT (3 iters)	87.00	31.80	69.80	68.00	48.00	69.25	58.00	55.79	68.00	63.68
SETS (3 iters, N=3)	86.60	31.60	**72.00**	68.25	**53.00**	29.50	67.86	56.84	71.00	59.08
TF-TTCL (N=3)	85.00	32.60	70.40	71.25	**53.00**	69.75	60.71	49.47	69.00	64.26
RM-Regen	**87.20**	**34.80**	71.40	**73.00**	52.00	**77.75**	**79.33**	**56.84**	**73.00**	**68.05**
Gemma 2-9B										
Self-Refine (3 iters)	40.00	32.80	27.80	28.00	16.00	54.25	24.00	41.05	59.00	35.77
ProCo (3 iters)	89.60	48.60	64.60	63.00	38.00	50.75	35.33	44.21	69.00	60.87
ST CoT (3 iters)	88.00	46.20	59.40	47.25	46.00	45.00	33.33	45.26	66.00	56.17
SETS (3 iters, N=3)	86.60	44.80	61.60	46.50	40.00	38.50	38.57	41.05	50.00	54.35
TF-TTCL (N=3)	88.80	47.40	60.00	50.50	44.00	49.50	**40.00**	49.47	67.00	58.25
RM-Regen	**89.80**	**49.40**	**66.20**	**66.00**	**48.00**	**66.00**	37.86	**51.58**	**71.00**	**64.84**

Table 4: Accuracy on nine benchmarks without oracle verification (verification-and-rectification baselines vs. ours). Iterative methods are evaluated with a maximum of three refinement iterations, while sampling-based methods use three sampled candidates (N = 3) for a fair comparison.

demonstrations provide more stable and informative guidance for regeneration than test-time distilled rules generated from imperfect predictions.

Figure 5 further presents the performance of iterative methods under different refinement budgets (2–5 iterations). We observe that under imperfect verification, increasing the number of iterative refinements does not consistently lead to better performance, and additional iterations may even introduce performance degradation. In contrast, our method outperforms all iterative baselines without relying on repeated verification-and-refinement loops, suggesting stronger robustness to verification noise and more effective utilization of verification feedback.

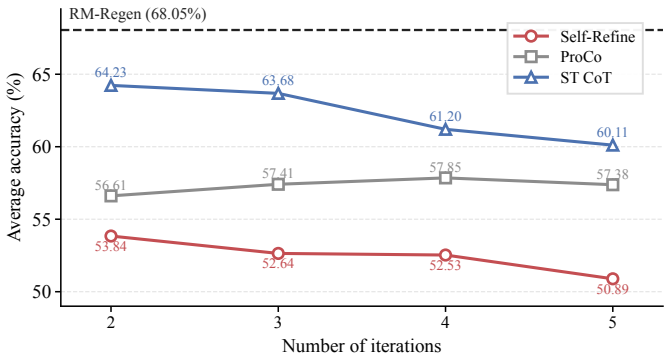


Figure 5: Average accuracy under noisy verification on Llama 3.1-8B. More refinement iterations do not consistently improve performance, while RM-Regen consistently outperforms all iterative baselines.

3.4 Ablation Study

We ablate our method along three axes: inference-stage modules, RM curation, and RM contents, to isolate the contribution of each design choice. Table 5 summarizes all variants on GSM_Hard with GPT-3.5. We first study inference configurations without oracle verification, and then analyze RM curation and RM contents under oracle verification. Below, we discuss each group in turn.

Variant	R	V	G	Acc.
Full (Ours)	✓	✓	✓	**64.00**
Fixed demos	✗	✓	✓	54.60
Direct boost	✓	✗	✗	44.60
Rectification	✓	✓	✗	53.60

(a) Inference-stage.

Variant	Ref.	Fil.	Tea.	Acc.
Full (Ours)	✓	✓	✓	**68.60**
w/o Refine	✗	✓	✓	63.20
w/o Filter	✓	✗	✓	66.40
w/o reflection	✓	✓	✗	56.80

(b) RM curation.

Variant	$\mathcal{R}^+$	$\mathcal{R}^-$	Prin.	Acc.
Full (Ours)	✓	✓	✓	**68.60**
w/o $\mathcal{R}^+$	✗	✓	✓	64.40
w/o $\mathcal{R}^-$	✓	✗	✗	64.20
w/o Principle	✓	✓	✗	67.80

(c) RM contents.

Table 5: GPT-3.5 on GSM_Hard: ablations of inference, RM curation, and RM contents. R/V/G: retrieval, verification, regeneration; Ref./Fil./Tea.: refine, filter, reflection; Prin.: principle.

Inference configuration Table 5a compares different inference configurations. Our full setting, combining retrieval, verification, and regeneration, achieves the best accuracy of 64.00%. Using fixed demonstrations without retrieval leads a substantial drop 9.4%, highlighting the importance of query-dependent retrieval. Direct performance boost without verification and regeneration performs the worst, suggesting that simply adding RM context is insufficient. Rectification instead of regeneration lags behind by 10.40%. These results confirm that retrieval, verification, and regeneration play complementary roles, and that our full configuration offers the best performance.

RM curation Table 5b shows that all three curation steps contribute to the final performance under oracle verification. Removing the reflection model causes the largest drop, from 68.60% to 56.80%, indicating that high-quality reflection feedback is crucial for constructing effective reflection memories. Disabling the ICL-oriented refine step leads to a moderate decrease of 5.4%, suggesting that adapting raw reflections and rectifications into an ICL-friendly format is important. Furthermore, skipping the filter is less harmful but still hurts performance 2.2%, implying that filtering primarily serves to remove low-quality or redundant memories and thus improves robustness rather than driving the main gains.

RM contents Table 5c examines which components of the RM are most beneficial. Removing either validate solution demonstrations $\mathcal{R}^+$ or reflections-and-rectification demonstrations $\mathcal{R}^-$ yields 4.2-4.4% drops, showing that both corrected and contrastive erroneous traces are necessary for strong in-context guidance. In contrast, removing distilled principles slightly degrades performance, indicating that principles provide an additional but secondary gain compared to explicit positive/negative examples.

4 Analysis

Beyond overall performance, we analyze the proposed method from four perspectives: dependency on reflection model; generalization across models and datasets; regeneration mechanism; and efficiency. These studies provide additional insight into when RM-Regen works, why it works, and how well it scales across models and tasks.

4.1 Reflection Model Dependency Analysis

To study the impact of reflection model, we construct RMs using reflection models of varying capabilities and evaluate performance with based model Llama 3.1-8B. In Table 6, GPT-5-mini, the strongest reflection model, achieves the highest accuracy across datasets, while weaker reflection models such as GPT-3.5, LLaMA-3.1-8B, and Gemma-2-9B cause slight drops, but still improve over few-shot CoT without a reflection. These results indicate that stronger reflection models provide better signals for RM construction, but overall performance does not critically depend on any specific reflection model. RMs generated by moderately capable models remain effective, capturing robust and transferable reasoning patterns.

reflection model	GSM-Hard	Bamboogle	Coin	avg%
No reflection (CoT)	31.80	51.00	65.75	47.30
GPT-5-mini	**41.00**	**63.00**	**85.75**	**61.10**
GPT-3.5	37.60	60.00	76.50	55.40
Llama 3.1-8B (self-reflect)	39.00	60.00	74.50	55.30
Gemma 2-9B	39.60	58.00	77.75	56.70

Table 6: reflection model dependency analysis. Performance evaluated with base model Llama 3.1-8B across different reflections.

4.2 Generalizability of RM Across Models and Datasets

Since RM construction incurs additional computational overhead, it is important to evaluate whether RM generalizes beyond the model and dataset used during its construction. We therefore examine two forms of out-of-domain transfer: (i) cross-model transfer, where an RM built for a base model is reused by different target models, and (ii) cross-dataset generalization, where RM signals derived from one dataset are applied to unseen tasks. As shown in Table 7, applying RM across different models or tasks leads to a modest performance drop, since the distilled reasoning principles originate from the source model's traces. Nevertheless, the transferred RM maintains strong performance across models and datasets, suggesting that it captures reusable reasoning patterns rather than overfitting to a specific model or task. These results indicate that a single pre-constructed RM can support multiple models and tasks, making the framework practical for large-scale deployment where rebuilding RM for every model or dataset would be costly.

	GSM8K	GSM-Hard	MATH
GPT-3.5	90.00 /90.20	51.20 /68.60	66.00 /63.40
Llama 3.1-8B	91.60 /92.60	35.00 /41.00	80.60 /80.60
Gemma 2-9B	91.40 /92.00	52.20 /53.00	71.20 /72.20

Table 7: Transferability of RM constructed for LLaMA-3.1-8B on MATH across models and datasets. Gray numbers denote native performance (the same model and dataset used for RM construction), while black numbers indicate cross-scenario performance under model or dataset shifts.

4.3 Regeneration Effectiveness Analysis

Regeneration is triggered when the verifier determines the output to be incorrect. Therefore, the regeneration rate is equivalent to the rate of predictions verified as incorrect.

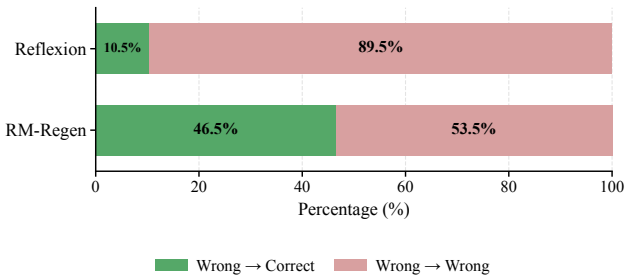
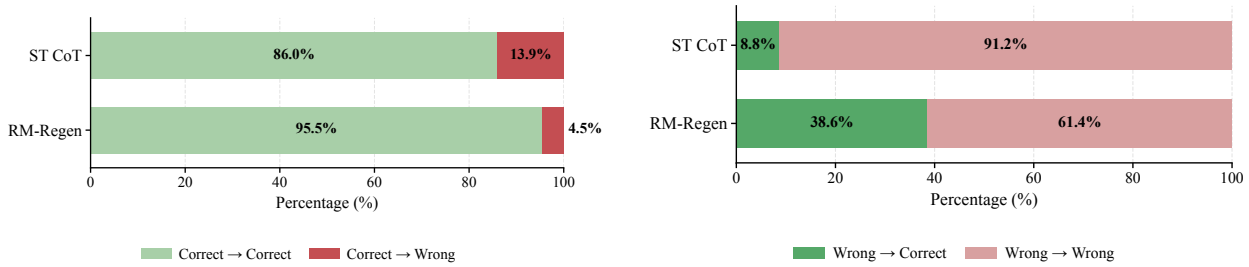


Figure 6: Wrong-to-correct transition of RM-Regen and Reflexion under oracle verification.

With Oracle Verifier When the verifier is an oracle, we analyze the rate of wrong-to-correct flips to better understand the effectiveness of our method. The evaluation of regeneration flip under an oracle verifier is shown in Figure 6. On the challenging GSM-Hard dataset, our method successfully rectifies 46.55% of the initial errors with the guidance of RM, substantially higher than the 10.47% achieved by Reflexion. This result highlights the importance of incorporating external guidance context, which helps the model correct errors that it could not reliably resolve through iterative self-refinement alone.



(a) Transitions among initially correct predictions.

(b) Transitions among initially incorrect predictions.

Figure 7: Correctness transitions of RM-Regen and ST CoT without oracle verification.

Without Oracle Verifier When the verifier is not an oracle, we analyze both wrong-to-correct and correct-to-wrong flips to evaluate the effectiveness and robustness of our method. The evaluation result is shown in Figure 7. Our method maintains a strong error-correction capability, achieving a 38.63% wrong-to-correct flip rate, 29.86% higher than ST CoT (4 iters). Regarding robustness, we observe a marginal 4.48% correct-to-wrong degradation, whereas ST CoT (4 iters) has 13.95% degradation. Overall, the improvement and robustness confirm that our framework provides a reliable reasoning safety net, even when the internal verifier is imperfect.

4.4 Regeneration vs. Rectification

The advantage of regeneration is reflected in comparisons with existing verification-and-rectification methods. Under oracle verification, RM-Regen consistently outperforms Reflexion across GPT-3.5, Llama-3.1-8B, and Gemma-2-9B, indicating that the gains arise not only from error detection but also from how failures are corrected.

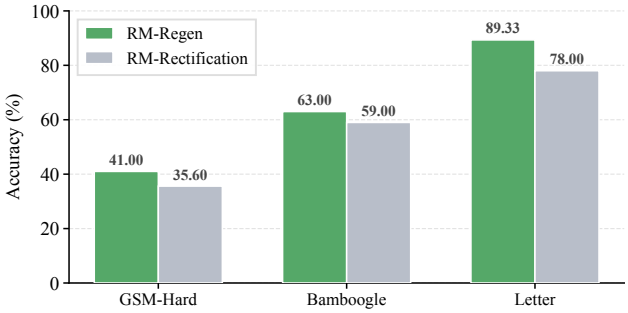


Figure 8: Comparison between regeneration and rectification with identical retrieved RM.

To directly compare regeneration and rectification, we evaluate an RM-Rectification variant while keeping the retrieved reflection memory fixed. In RM-Rectification, the model uses the retrieved RM to generate a task-specific rectification analysis and then revises the previous answer. In RM-Regen, the model discards the previous trajectory and regenerates a fresh answer guided by RM. As shown in Figure 8, regeneration consistently achieves higher accuracy across all three benchmarks, with gains of up to 11.33 % points. This suggests that discarding an unreliable reasoning trajectory and regenerating from retrieved guidance is more effective than locally revising the original reasoning process, consistent with recent findings on contextual drag, where erroneous reasoning traces can bias subsequent reasoning toward similar errors Cheng et al. (2026).

4.5 Computational Cost Analysis

Our framework consists of a one-time offline cost for constructing reflection memory (RM) and a per-query online inference cost. Figure 9 summarizes the online speed–accuracy trade-off of verification-guided

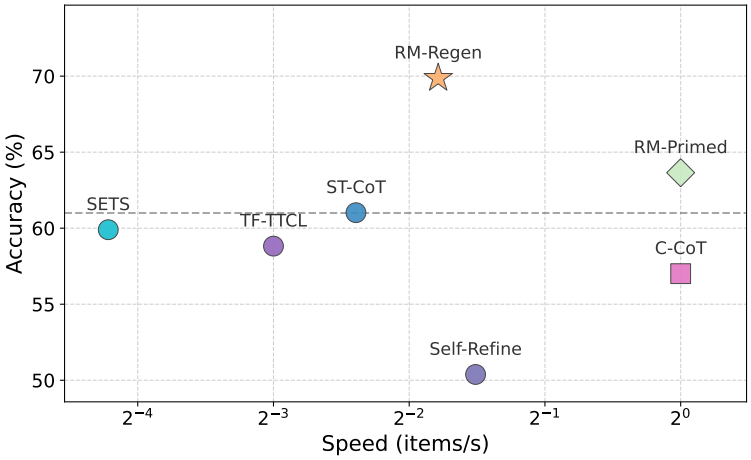


Figure 9: Speed–accuracy trade-off of verification-guided reasoning improvement methods on GPT-3.5. For iterative methods, the maximum number of refinement iterations is set to three, while SETS and TF-TTCL use three sampled candidates. RM-Regen achieves the highest accuracy while maintaining competitive inference speed.

methods. Iterative baselines (Self-Refine and ST-CoT) and sampling baselines (SETS and TF-TTCL) lie in the lower-left region, indicating slower inference with only moderate gains. As refinement iterations increase, speed drops while accuracy improvements are often marginal or even negative. By contrast, RM-Primed and RM-Regen lie in the upper-right region, achieving higher accuracy at higher speed. Although Contrastive-CoT is slightly faster than Self-Refine and ST-CoT, its accuracy remains clearly below ours. We now quantify (i) offline RM construction cost, (ii) online inference cost, and (iii) amortized total cost.

Offline RM construction cost. The offline cost is: $C_{\text{offline}} = kN \cdot C_{\text{base}} + M \cdot C_{\text{reflection}}$, where N is the number of training questions for RM construction, k is the number of base-model samples per question used to collect errors/inconsistencies, C_{base} is the cost of one base-model generation, M is the number of selected failed/inconsistent cases used as RM entries, and $C_{\text{reflection}}$ is the cost of reflection-model call for one RM construction. Importantly, reflection-model calls are only required during RM construction and are never used at inference time.

Online inference cost. RM-Primed uses a single base-model generation at inference time. RM-Regen uses a small number of RM-guided verification calls (eg. 3) followed by one regeneration call (if triggered). Both are more efficient than iterative rectification baselines, whose cost increases with refinement iterations.

Amortized total cost. The per-query total cost over Q downstream queries is: $C_{\text{total}}(Q) = \frac{C_{\text{offline}}}{Q} + C_{\text{online}}$. For large Q, C_{offline}/Q becomes small. Therefore, RM-Regen retains its online speed–accuracy advantage while the one-time offline cost is amortized in large-scale serving scenarios.

5 Related Work

ICL Demonstrations Improves Reasoning Well-designed few-shot demonstrations can boost LLM performance. Auto-CoT Zhang et al. (2023) automatically generates CoT examples and, with proper selection, can outperform hand-crafted CoT. Building on this, contrastive CoT Chia et al. (2023) and related methods Mo et al. (2024); Gao & Das (2024) leverage paired correct and incorrect reasoning examples, often accompanied by error analyses, to help models avoid common reasoning failures. Gu et al. (2025) further extends this paradigm by retrieving relevant contrastive experiences from memory to support reasoning. More broadly, memory-augmented approaches have been explored to improve LLM capabilities. For example, MemLLM Modarressi et al. (2025) equips LLMs with an explicit read–write memory for storing and retrieving factual knowledge, and TF-TTCL Zheng et al. (2026) improves generation accuracy via online rule distillation from contrastive trajectories and retrieval-guided inference. Unlike these approaches, which

primarily retrieve demonstrations or task knowledge, our reflection memory stores structured verification, reflection, correction, and principle information derived from past reasoning failures. As a result, RM serves not only as an example repository for in-context learning, but also as a reusable source of corrective knowledge that supports both RM-guided verification and RM-guided regeneration.

Self-Reflection and Self-Refinement Leveraging LLMs' self-assessment and refinement abilities has emerged as a promising direction. Several training-free methods have been proposed: Self-Refine Madaan et al. (2023) iteratively refines outputs via self-assessment; Reflexion Shinn et al. (2023) lets LLMs verbally reflect on feedback and store reflections in memory buffer to guide future trials; ST CoT Wu et al. (2025) shows that iteratively reusing model-generated information can reduce prediction uncertainty in a self-training–style framework. More recently, SETS Chen et al. (2025) explores test-time scaling through sampling, self-verification, and self-correction, demonstrating that verification signals can effectively guide additional inference-time computation. For small-scale LLMs (SLLMs), fine-tuning–based approaches have also been explored. ReflectEvo Li et al. (2025) builds a reflection learning dataset and improves SLLM reasoning via SFT and DPO. SCFT and RLERR Wang et al. (2026) enhance reflective reasoning through self-critique fine-tuning and reflection-oriented reinforcement learning. Recent work has further highlighted the importance of verification. Learning to Self-Verify Chen et al. (2026) shows that self-verification constitutes a capability distinct from generation and that explicitly learning to self-verify improves downstream reasoning performance. Retrieval-based verification has also been explored for hallucination prevention, for example Premise Verification Qin et al. (2026) verify the validity of premises in user queries through retrieval-augmented logical reasoning before generation. Despite recent progress, existing approaches either rely solely on online self-generated feedback, which is prone to error accumulation, or require additional training to improve reflection quality and correction effectiveness. In contrast, we organize corrective reflection knowledge into a reusable reflection memory that can be constructed offline and retrieved at inference time to support both RM-guided verification and RM-guided regeneration, without parameter updates.

Beyond explicit verification, recent work has explored reinforcement learning and test-time optimization to improve the verify–improve loop, including TTRL Zuo et al. (2025), RESTRAIN Yu et al. (2025), and evolution-based selection methods Zhou et al. (2025). Unlike these approaches, our method improves reasoning through memory-guided verification and regeneration, without additional training, online optimization, or excessive sampling.

6 Conclusion

This paper proposes a training-free regeneration paradigm in which Reflection Memory (RM) provides contextual guidance to correct reasoning errors and regeneration from scratch enables break out of faulty reasoning loops. Under oracle verification, our method achieves the best performance and remains the most robust with noisy verification. Without oracle verification, RM–guided self-verification attains the highest F1 score, reliably triggering regeneration. Overall, this paradigm delivers significant gains in LLM reasoning performance while maintaining low computational cost.

7 Limitations

Although our training-free regeneration paradigm achieves strong performance across diverse tasks, it still has several limitations that need further investigation. First, the quality of RM is highly dependent on the reflection model. Without a sufficiently powerful reflection model to generate reliable reflections, summarize principles, and provide accurate rectifications, the effectiveness of RM can degrade substantially. Although our verifier attains high recall, it is still imperfect, and some incorrect responses remain undetected. These missed errors are a bottleneck that limits the overall performance ceiling. Future work could improve RM quality through multi-reflection aggregation or distillation, and develop more reliable RM-aware verifiers, such as lightweight learned or ensemble-style models that leverage logged missed errors for offline calibration. These directions may further reduce undetected mistakes while largely preserving the single-pass regeneration paradigm.

Broader Impact Statement

This work aims to improve LLMs' performance through response verification and rectification. However, imperfect verification can still miss errors or incorrectly modify correct answers, potentially leading to overconfidence or unnecessary rewrites. In addition, the same verification and rewriting mechanisms could be misused to produce more persuasive but incorrect content; therefore, we recommend cautious use in safety-critical settings and emphasize the need for human oversight. We do not collect new personal data, and all experiments are conducted on publicly available benchmarks.

References

Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. Forest-of-thought: Scaling test-time compute for enhancing llm reasoning. In *Proceedings of the 42nd International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2025.

Jiefeng Chen, Jie Ren, Xinyun Chen, Chengrun Yang, Ruoxi Sun, Jinsung Yoon, and Sercan O Arik. SETS: Leveraging self-verification and self-correction for improved test-time scaling. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. In *The Twelfth International Conference on Learning Representations*, 2024.

Yuxin Chen, Yu Wang, Yi Zhang, Ziang Ye, Zhengzhou Cai, Yaorui Shi, Qi Gu, Hui Su, Xunliang Cai, Xiang Wang, et al. Learning to self-verify makes language models better reasoners. *arXiv preprint arXiv:2602.07594*, 2026.

Yun Cheng, Xingyu Zhu, Haoyu Zhao, and Sanjeev Arora. Contextual drag: How errors in the context affect llm reasoning. *arXiv preprint arXiv:2602.04288*, 2026.

Yew Ken Chia, Guizhen Chen, Luu Anh Tuan, Soujanya Poria, and Lidong Bing. Contrastive chain-of-thought prompting. *arXiv preprint arXiv:2311.09277*, 2023.

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. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Jiaxun Gao, Him Wai Ng, and Z. Jane Wang. Once-more: Continuous self-correction for large language models via perplexity-guided intervention. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=3CKdjb5SuH`.

Xiang Gao and Kamalika Das. Customizing language model responses with contrastive in-context learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 18039–18046, 2024.

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

Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*, 2024.

Jiawei Gu, Ziting Xian, Yuanzhen Xie, Ye Liu, Enjie Liu, Ruichao Zhong, Mochi Gao, Yunzhi Tan, Bo Hu, and Zang Li. Toward structured knowledge reasoning: Contrastive retrieval-augmented generation on experience. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 23891–23910, 2025.

Neel Guha, Julian Nyarko, Daniel Ho, Christopher Ré, Adam Chilton, Aditya K, Alex Chohlas-Wood, Austin Peters, Brandon Waldon, Daniel Rockmore, Diego Zambrano, Dmitry Talisman, Enam Hoque, Faiz Surani, Frank Fagan, Galit Sarfaty, Gregory Dickinson, Haggai Porat, Jason Hegland, Jessica Wu,

Joe Nudell, Joel Niklaus, John Nay, Jonathan Choi, Kevin Tobia, Margaret Hagan, Megan Ma, Michael Livermore, Nikon Rasumov-Rahe, Nils Holzenberger, Noam Kolt, Peter Henderson, Sean Rehaag, Sharad Goel, Shang Gao, Spencer Williams, Sunny Gandhi, Tom Zur, Varun Iyer, and Zehua Li. Legalbench: A collaboratively built benchmark for measuring legal reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 44123–44279, 2023.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. V-star: Training verifiers for self-taught reasoners. In *Conference on Language Modeling*, 2024.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*, 2024a.

Sili Huang, Jifeng Hu, Hechang Chen, Lichao Sun, and Bo Yang. In-context decision transformer: reinforcement learning via hierarchical chain-of-thought. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 19871–19885, 2024b.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*, 2021.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When can llms actually correct their own mistakes? a critical survey of self-correction of llms. *Transactions of the Association for Computational Linguistics*, 12:1417–1440, 2024.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahan, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Dongseok Lee, Jimyung Hong, Dongyoung Kim, and Jaehyung Kim. Training-free llm verification via recycling few-shot examples. *arXiv preprint arXiv:2506.17251*, 2025.

Jiaqi Li, Xinyi Dong, Yang Liu, Zhizhuo Yang, Quansen Wang, Xiaobo Wang, SongChun Zhu, Zixia Jia, and Zilong Zheng. Reflectevo: Improving meta introspection of small llms by learning self-reflection. In *Findings of the Association for Computational Linguistics (ACL) 2025*, 2025.

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. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594, 2023.

Ying Mo, Jiahao Liu, Jian Yang, Qifan Wang, Shun Zhang, Jingang Wang, and Zhoujun Li. C-ICL: Contrastive in-context learning for information extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 10099–10114, 2024.

Ali Modarressi, Abdullatif Köksal, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schuetze. MemLLM: Fine-tuning LLMs to use explicit read-write memory. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL `https://openreview.net/forum?id=dghM7sOudh`.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023.

Yuehan Qin, Li Li, Yi Nian, Xinyan Velocity Yu, Yue Zhao, and Xuezhe Ma. Don't let it hallucinate: Premise verification via retrieval-augmented logical reasoning. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856.

Bilgehan Sel, Ahmad Al-Tawaha, Vanshaj Khattar, Ruoxi Jia, and Ming Jin. Algorithm of thoughts: Enhancing exploration of ideas in large language models. In *Proceedings of the 41st International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2024.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. Synthetic prompting: Generating chain-of-thought demonstrations for large language models. In *International Conference on Machine Learning*, pp. 30706–30775. PMLR, 2023.

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

Mo Shu, Zihao Zhong, Zhitao Xie, Baolin Peng, and Wenhu Chen. Hard gsm8k: A more challenging benchmark for llm reasoning. *arXiv preprint arXiv:2402.03301*, 2024.

Ankur Sinha and Tanmay Khandait. Impact of news on the commodity market: Dataset and results. In *Advances in Information and Communication: Proceedings of the 2021 Future of Information and Communication Conference (FICC), Volume 2*, pp. 589–601. Springer, 2021.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. In *Advances in Neural Information Processing Systems*, volume 33, pp. 16857–16867, 2020.

Guiyao Tie, Zenghui Yuan, Zeli Zhao, Chaoran Hu, Tianhe Gu, Ruihang Zhang, Sizhe Zhang, Junran Wu, Xiaoyue Tu, Ming Jin, Qingsong Wen, Lixing Chen, Pan Zhou, and Lichao Sun. Can LLMs correct themselves? a benchmark of self-correction in LLMs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

Hanbin Wang, Jingwei Song, Jinpeng Li, Qi Zhu, Fei Mi, Ganqu Cui, Yasheng Wang, and Lifeng Shang. Teaching large reasoning models effective reflection. *arXiv preprint arXiv:2601.12720*, 2026.

Haozhe Wang, Haoran Que, Qixin Xu, Minghao Liu, Wangchunshu Zhou, Jiazhan Feng, Wanjun Zhong, Wei Ye, Tong Yang, Wenhao Huang, Ge Zhang, and Fangzhen Lin. Reverse-engineered reasoning for open-ended generation. *arXiv preprint arXiv:2509.06160*, 2025.

Xuezhi Wang and Denny Zhou. Chain-of-thought reasoning without prompting. In *Advances in Neural Information Processing Systems*, volume 37, pp. 66383–66409, 2024.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022.

Zhenyu Wu, Qingkai Zeng, Zhihan Zhang, Zhaoxuan Tan, Chao Shen, and Meng Jiang. Large language models can self-correct with key condition verification. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 12846–12867, 2024.

Zongqian Wu, Baoduo Xu, Ruochen Cui, Mengmeng Zhan, Xiaofeng Zhu, and Lei Feng. Rethinking chain-of-thought from the perspective of self-training. In *Forty-second International Conference on Machine Learning*, 2025.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822, 2023.

Jiaxuan You, Mingjie Liu, Shrimai Prabhumoye, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Llm-evolve: Evaluation for llm's evolving capability on benchmarks. In *Proceedings of the 2024 conference on Empirical Methods in Natural Language Processing*, pp. 16937–16942, 2024.

Zhaoning Yu, Will Su, Leitian Tao, Haozhu Wang, Aashu Singh, Hanchao Yu, Jianyu Wang, Hongyang Gao, Weizhe Yuan, Jason Weston, et al. Restrain: From spurious votes to signals–self-driven rl with self-penalization. *arXiv preprint arXiv:2510.02172*, 2025.

Tianjun Zhang, Aman Madaan, Luyu Gao, Steven Zhang, Swaroop Mishra, Yiming Yang, Niket Tandon, and Uri Alon. In-context principle learning from mistakes. In *ICML 2024 Workshop on In-Context Learning*, 2024.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*, 2023.

Kaiwen Zheng, Kai Zhou, Jinwu Hu, Te Gu, Mingkai Peng, and Fei Liu. Training-free test-time contrastive learning for large language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 29644–29676, 2026.

Yujun Zhou, Zhenwen Liang, Haolin Liu, Wenhao Yu, Kishan Panaganti, Linfeng Song, Dian Yu, Xiangliang Zhang, Haitao Mi, and Dong Yu. On the evolution of language models without labels: Majority drives selection, novelty promotes variation. In *The 5th Workshop on Mathematical Reasoning and AI at NeurIPS 2025*, 2025.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. TTRL: Test-time reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

A Implementation Details

	Algorithmic			Reasoning		Symbolic		Domain	
Dataset	GSM8K[1]	GSM-Hard[2]	MATH[3]	StrategyQA[4]	Bamboogle[5]	Coin[6]	Letter[7]	LB[8]	Headline[9]
Train split	800	800	800	800	10	100	10	5	100
Test split	500	500	500	400	100	400	140	95	100
Answer type	Num.	Num.	Num.	Bool.	Factoid	Bool.	Letter	MCQ	MCQ

Table 8: Dataset splits used in our experiments. For each benchmark, we report the number of train and test examples used for reflection-memory construction and evaluation. We randomly subsample train/test examples from each benchmark using a fixed random seed, ensuring the same subset across all methods.

	Algorithmic			Reasoning		Symbolic		Domain	
Dataset	GSM8K	GSM-Hard	MATH	StrategyQA	Bamboogle	Coin	Letter	LB	Headline
GPT-5mini	94.00	75.00	97.80	80.75	66.00	79.50	96.67	84.21	79.00

Table 9: Performance of reflection model on nine benchmarks.

A.1 Dataset Details and Splits

The evaluation covers nine benchmarks spanning algorithmic, reasoning, symbolic, and domain-specific tasks, providing a diverse test cases for assessing both general problem-solving ability and specialized-domain reliability. These benchmarks vary in input formats, reasoning requirements, and answer types, allowing us to examine the robustness of each method across heterogeneous settings rather than on a single task family. The corresponding dataset statistics, including the number of examples and the evaluation splits used in our experiments, are summarized in Table 8. All datasets are obtained from publicly available sources; license and usage terms follow the corresponding dataset cards and original releases.

A.2 Inference and Hyperparameter Settings

We evaluate both proprietary and open-source LLMs. Specifically, we query GPT-3.5-0125 and GPT-5-mini-2025-08-07 via the OpenAI API, while LLaMA 3.1-8B-Instruct, Gemma 2-9B-it is deployed locally using vLLM on an NVIDIA A6000 GPU for efficient batched inference.

Inference-time decoding details We use a unified decoding configuration across all benchmarks and methods to ensure a controlled comparison. When multiple generations are required (e.g., sampling-based verification), we adopt sampling with temperature = 0.3 and top_p = 0.7, and cap the maximum generation length at 1024 tokens (max_tokens = 1024). For verification-based components, we draw three independent samples per query under the same sampling settings and feed the resulting verification outcomes into the corresponding verification pipeline. When only a single generation is needed, we switch to deterministic decoding to improve stability and reproducibility, using temperature = 0 and top_p = 1 (with the same `max_tokens` limit). All other inference parameters are kept at their default values for the OpenAI API and vLLM, respectively.

[1] https://huggingface.co/datasets/gsm8k
[2] https://huggingface.co/datasets/reasoning-machines/gsm-hard
[3] https://huggingface.co/datasets/nlile/hendrycks-MATH-benchmark
[4] https://huggingface.co/datasets/tasksource/strategy-qa
[5] https://huggingface.co/datasets/chiayewken/bamboogle
[6] https://huggingface.co/datasets/skrishna/coin_flip
[7] https://huggingface.co/datasets/ChilleD/LastLetterConcat
[8] https://huggingface.co/datasets/nguha/legalbench
[9] https://huggingface.co/datasets/SaguaroCapital/sentiment-analysis-in-commodity-market-gold

Retrieval and reranking hyperparameters For the retrieval-then-reranking pipeline, we select the number of retrieved candidates n and the number of reranked memories k based on a trade-off between retrieval coverage and inference efficiency. A larger n improves the likelihood of recalling relevant memories at the cost of additional computation, while a smaller k helps reduce noise and stabilize in-context learning. In our experiments, we fix $n = 10$ and $k = 3$ across all benchmarks, as these values provided stable performance. We found that moderate variations around these settings lead to similar trends, indicating that our method is not overly sensitive to the exact choice of n and k.

A.3 Performance of the reflection model

We report the reflection model's Few-shot CoT performance in Table 9. Overall, the reflection model achieves consistently strong accuracy across the evaluated benchmarks, indicating that it can serve as a reliable source for generating RM (including reflection, principle, and rectification) in our pipeline. In particular, its high correctness rate suggests that the produced reflections are more likely to identify the true failure modes of incorrect responses, and that the rectified demonstrations provide accurate, task-relevant reasoning patterns. This is important because the quality of both reflection and correction is bounded by the reflection model's competence; a sufficiently strong reflection model reduces noisy feedback and improves the usefulness of the resulting reflection memories.

A.4 Reproduction Details of Baselines

Training-based baselines. We treat ReflectEvo[10] as a *training-based* baseline. ReflectEvo considers four training configurations: (i) one-stage training of both self-reflection and self-correction on D^{+}; (ii) two-stage training of self-reflection and self-correction on D^{+}; (iii) self-reflection only on $D^{\pm}$; and (iv) self-reflection only on D^{pref}. In our experiments, we reproduce the two-stage w/ D^{+} setting, which achieves the best results in the LogiQA task in the original paper. For algorithmic tasks (GSM8K, GSM-Hard, and MATH), we fine-tune Llama-3.1-8B using the `math` training data released by ReflectEvo. For the remaining benchmarks, we fine-tune Llama-3.1-8B on the `logiqa` training data to assess the generalizability of ReflectEvo beyond its original training domain.

Training-free prompt-based baselines. We treat Self-Refine[11], Reflexion[12], ST-CoT[13], ProCo[14], and Contrastive-CoT[15] as *training-free prompt-based* baselines, since they do not update model parameters and differ only in their prompting and inference procedures. For each baseline, we carefully reproduce the core algorithm in our unified evaluation framework: we read the original paper and public implementation, extract the key inference steps, and re-implement them on top of our codebase while keeping the underlying LLM, decoding settings, and evaluation splits identical across methods. We follow the original prompt templates as closely as possible and adapt only the task-specific content. For each benchmark, we construct a fixed set of few-shot demonstrations from the training split; unless otherwise specified, all methods use the same set and number of in-context examples on a given dataset. For iterative baselines, we evaluate multiple numbers of refinement iterations, and report the corresponding results in Sec. 3.

B Prompts

This section presents the prompt templates used in our method, including prompts for reasoning-quality scoring, response reflection, response verification, and principle extraction. These templates standardize the evaluation and refinement steps, ensuring consistent judgments and enabling reliable construction and retrieval of reflection memories across tasks.

[10] `https://github.com/bigai-nlco/ReflectEvo`
[11] `https://github.com/madaan/self-refine`
[12] `https://github.com/noahshinn/reflexion`
[13] `https://github.com/zongqianwu/ST-COT`
[14] `https://github.com/wzy6642/ProCo`
[15] `https://github.com/DAMO-NLP-SG/contrastive-cot`

B.1 Principle Prompt

Developer: You are an Insight Extraction Agent.

Given:
Reflection: {analysis} (contains detailed error analysis and corrective reasoning)

Instructions:

- Focus on deriving a concise, generalizable principle that can improve future reasoning performance.
- Do not restate or re-analyze the reflection; instead, abstract its key lessons into clear, reusable insights.
- Make the principle clear and actionable.
- The principle should be under 20 words.

Output Format:
<one concise principle or guideline that should be emphasized to improve future performance>

B.2 Reflection Prompt

Developer: You are a precise tutor. Analyze the errors in the original reasoning, then provide a concise and corrected reasoning.

Given:
Question: {question}
Incorrect Reasoning: {prediction}
Gold Reasoning for Reference: {oracle}

Instructions:

- In `Analysis`: Briefly list the main error(s) in the original reasoning (1–3 bullet points max).
- In `Corrected Reasoning`:
  1. Do **not** copy the original reasoning; fix it.
  2. Number the steps; include key equations and unit/constraint checks.
  3. Keep reasoning concise (no redundant explanations).
  4. Keep the final answer exactly equal to Gold Reasoning.
  5. Do not copy text verbatim from the Original Reasoning; rewrite with corrections.

Output Format:
Analysis: - <key mistake(s) >
Corrected Reasoning: <revised reasoning >#### <final answer >

B.3 Verification Prompt

Developer: You are an answer verification agent. Your task is to determine whether the provided answer is numerically and logically correct with respect to the question.

Instructions:

- Extract the final answer.
- Briefly check whether the answer satisfies the numeric or logical constraints of the question.
- Focus on correctness, not writing style or explanation quality.

Here are some assessment examples:
Each example includes a question, an answer, an analysis of the answer, and a final assessment ("correct" or "incorrect"). Learn from these examples how to assess the correctness of reasoning.

{examples}

Now assess the following case (do not repeat the answer in the analysis):
Question: {question}
Answer: {answer}

Output Format:
Analysis: <brief reason >
Assessment: <correct/ incorrect >

B.4 ICL Oriented Refine Prompt

Developer: You are a careful wrong example cleaner for contrastive In-Context Learning.
Contrastive In-Context Learning works by contrasting the wrong reasoning with the corrected reasoning. Your task is to rewrite a noisy or verbose wrong example into a SHORT, CLEAR negative example that highlights exactly the mistake described in the Analysis.

Given:
Question: {question}
Incorrect Reasoning: {prediction}
Analysis of the Mistake: {analysis}
Corrected Reasoning: {corrected}

Instructions:

- Keep mistakes indicated in Analysis and preserve the WRONG final answer if present.
- Be as concise as the corrected reasoning.
- No new variables, new inequalities, no new algebraic chains, no filler text, no paraphrasing.
- Do NOT include any explanation or meta text—output ONLY the cleaned negative example.
- If there is a wrong result, end the output with '#### <final answer >'.

Output Format:
Return ONLY the cleaned negative example with the wrong conclusion (if the original had one). End with the wrong answer in the format '#### <final answer >' if applicable.

B.5 Score Reasoning Quality Prompt

Developer: You are a strict grader for reasoning quality. Evaluate if the reasoning is high-quality for use as an ICL demonstration.

Given:
Question: {question}
Reasoning: {prediction}

Instructions:
Score the original reasoning strictly in [1–10] based on:

- Logic validity: reasoning is logically sound, no gaps.
- Arithmetic & units: calculations and units are correct.
- Completeness & checkability: key steps are covered and can be verified.
- Brevity & clarity: concise and clear without redundancy, similar to the gold reasoning.
- Format compliance: proper step-by-step to final answer structure.

Scoring anchors:

- 1–3 = unusable (major errors, misleading)
- 4–6 = partially usable (some errors but partial reasoning correct)
- 7–8 = good (minor flaws but mostly correct and clear)
- 9–10 = excellent (fully correct, concise, clear, ICL-ready)

Output strictly a single integer (1–10). Do not add any words, symbols, or explanations.