
A Stitch in Time Saves Nine: Proactive Self-Refinement for Language Models

Anonymous Author(s)

Affiliation

Address

email

Abstract

Recent advances in self-refinement have demonstrated significant potential for improving the outputs of large language models (LLMs) through iterative refinement. However, most existing self-refinement methods rely on a reactive process with a fixed number of iterations, making it difficult to determine the optimal timing and content of refinement based on the evolving generation context. Inspired by the way humans dynamically refine their thoughts during execution, we propose ProActive Self-Refinement (PASR), a novel method that enables LLMs to refine their outputs during the generation process. Unlike methods that regenerate entire responses, PASR proactively decides whether, when, and how to refine based on the model’s internal state and evolving context. We conduct extensive experiments on a diverse set of 10 tasks to evaluate the effectiveness of PASR. Experimental results show that PASR significantly enhances problem-solving performance. In particular, on Qwen3-8B, PASR reduces average token consumption by 41.6% compared to standard generation, while also achieving an 8.2% improvement in accuracy. Our code and all baselines used in the paper are available in the GitHub <https://anonymous.4open.science/status/Proactive-Self-Refine-in-LLMs-CB05>.

1 Introduction

Self-refinement, as a fundamental cognitive capacity, is essential for effective problem-solving in both humans and models. Self-refinement is characterized by the active monitoring of one’s thought processes, the identification and subsequent remediation of errors, and the iterative adjustment of responses and behaviors [1][2]. Its significance in human intelligence highlights a promising direction for developing more autonomous and robust AI agents [3][4][5]. Inspired by this powerful cognitive process, recent work has applied the self-refinement to Large Language Models (LLMs).

Existing LLM self-refinement methods typically follow **patch-after-failure (post-hoc)** paradigm, where an initial response is generated and then iteratively improved based on feedback through multiple rounds of refinement iterations[6][7][8][9]. Broadly, these methods fall into two categories. The first employs carefully crafted prompts to elicit self-refinement behaviors, often by explicitly instructing it to correct or refine its previous outputs [10][11][12]. The second leverages supervised fine-tuning on synthetic datasets that pair suboptimal responses with improved versions, training the model to refine its outputs automatically [13][14].

While these post-hoc self-refinement methods have demonstrated the performance gains across various tasks, they still lack the ability to proactively determine **whether, when and how** to perform refinement. (*Whether:*) these methods are often applied in a blind, static manner after initial generation, regardless of whether refinement is necessary. This delayed intervention frequently requires multiple iterative steps to yield meaningful improvement, yet the optimal number of iterations is neither predefined nor easily inferred, often requiring extensive empirical tuning [14][6]. (*When:*)

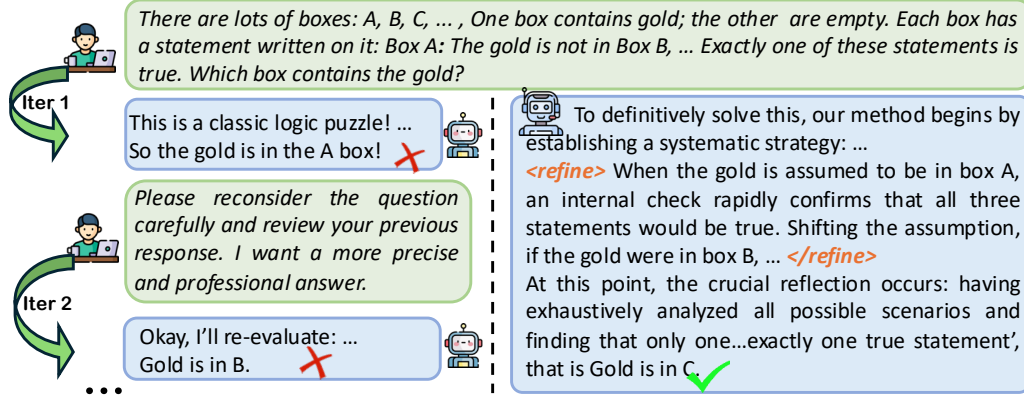


Figure 1: Comparison between the post-hoc refinement method (left) and our proposed PASR (right). The post-hoc refinement method iteratively refines its initial answer. In contrast, PASR proactively refines its reasoning process during the generation.

once an error or deviation is introduced during the initial generation and is not properly addressed, it can propagate throughout subsequent steps [15][16], making effective recovery significantly more challenging. (*How:*) these methods also rely heavily on external feedback mechanisms, such as tool-assisted evaluations and auxiliary models [17][4][18], to identify and correct errors. [8] demonstrates that without appropriate external feedback, self-refinement loops even lead to performance degradation.

We argue that it is indispensable to enhance the capability of LLMs to perform **proactive self-refinement** during the generation process, enabling models to autonomously determine the appropriate timing and content for refinement based on the evolving context. However, even advanced LLMs equipped with strong deep thinking capabilities, such as DeepSeek R1 [19] and O1¹, still struggle to achieve satisfactory proactive self-refinement. Although their reasoning process involve various meta-cognitive functions such as planning [20][21] and evaluation [22][23], they *lack a dedicated mechanisms optimized for proactive self-refinement*. As a result, these models often bring superficial self-refinement [24] and fall into cognitive dilemmas, exhibiting patterns of overthinking [25][26] and underthinking [27], as widely observed in recent studies.

To equipping the model with such proactive self-improvement capability, a straightforward method is to perform supervised training on data that demonstrates adaptive refinement behavior. However, this method faces two significant challenges. First, constructing such demonstration data is non-trivial, as it is impractical to define criteria for the optimal timing for refinement during the generation. It is impractical to distill from advanced LLMs. Furthermore, simply imitating such data is insufficient for the model to truly acquire this capability [28][29]. The model struggles to generalize adaptive self-refinement behavior to unseen tasks, and in some instances, its performance even degrades.

Therefore, we propose **ProActive Self-Refinement (PASR)**, a Reinforcement Learning (RL) method to train LLMs to refine their outputs adaptively during generation. The difference between pos-hoc refinement and PASR is show in Figure 1. PASR leverages on-policy rollouts from the learner model to explore whether, when and how to refine, conditioned on the task and generation state, rather than relying on predefined rules or manually designed refinement positions. Different from supervised learning, RL relies heavily on the reward signals to shape the model’s behavior [30][31]. A key challenge is defining what counts as an effective refinement. If the rewards are misaligned, the model may either miss important opportunities for refinement or make unnecessary refinements to already good outputs. Therefore, we introduce a proxy evaluation strategy that compares the refinements relative to the standard outputs, encouraging timely, necessary, and contextually appropriate refinement.

In summary, our main contributions are summarized as follows: (1) To the best of our knowledge, we are the first to propose enhancing proactive self-refinement as a formal task, aiming to equip LLMs with the ability to refine their outputs in an dynamic and self-directed manner during generation. (2) We propose PASR, a method that enables proactive self-refinement throughout the generation process via reinforcement learning. (3) We design a comparison-based reward strategy to assess

¹<https://openai.com/o1/>

the effectiveness of proactive self-refinement and guide model behavior during training. (4) We empirically demonstrate the effectiveness and efficiency of PASR across a diverse set of tasks. In particular, on Qwen3-8B, PASR significantly reduces average token consumption by 41.6% compared to the standard generation method, while also achieving a 8.2% improvement in accuracy.

2 Method

2.1 Task Formulation

Unlike existing post-hoc refinement methods, our task is that empowers the model to proactively self-refine its generated content during the generation process. We formalize this in-process refinement behavior as follows:

Error Correction. Fixing factual inaccuracies, logical fallacies, or computational mistakes introduced in earlier outputs.

Information Complement. Filling in missing yet critical details to ensure completeness and correctness.

Solution Improvement. Improving the effectiveness and efficiency of the proposed solution by introducing more advanced strategies or refined representations.

Task Alignment. Re-aligning content with the task goal or user intent when divergence is detected.

The model proactively decides whether, when and how to refine previously generated parts of its internal reasoning trace, integrating these updates into its ongoing generation process. This sequential decision-making problem is naturally formulated as a Markov Decision Process (MDP) [32].

Formally, given an input query x , the goal is to generate a final response y' . This is achieved through an iterative refinement process that constructs an intermediate generation trace $z = (z_1, z_2, \dots, z_T)$, where T is the total number of generation tokens. At each timestep i (from 1 to T), the model is in the **state** s_i , which is determined by the input x and the trace generated $z_{\{1:i-1\}}$ so far. It then takes an **action** a_i chosen from an action space \mathcal{A} , which consists of two main types of actions: **Content Generation** a_{gen} and **Trace Refinement** a_{refine} . The **Content Generation** extends the current trace $z_{\{1:i-1\}}$ by appending a new token, segment, or reasoning step, resulting in $z_{\{1:i\}}$. The **Trace Refinement** revises a specific portion of the existing trace $z_{\{1:i-1\}}$ to improve its quality. This modified trace also forms $z_{\{1:i\}}$. The sequence of states, actions, and resulting trace segments $((s_1, a_1, z_1), \dots, (s_T, a_T, z_T))$ constitutes an **observation**. The final response y' is derived from the complete trace z . The training objective is to learn the optimal **policy** π that maximizes the expected reward of proactive refinement responses. The reward, denoted as $R_{y'}$, reflects the quality of the response resulting from proactive trace refinement. The objective is formalized as:

$$\max_{\pi} \sum_x \mathbb{E}_{y' \sim \pi(\cdot|x)} [R_{y'}] \quad (1)$$

2.2 PASR: ProActive Self-Refinement via RL

In this work, we employ Group Reference Policy Optimization (GRPO) algorithm, a variant of Proximal Policy Optimization (PPO), specifically designed to stabilize learning through group-wise advantage normalization. By normalizing advantages within groups of responses generated from the same input, GRPO reduces variance in the policy gradient updates and promotes stable learning. Let π_{θ} represent the current policy parameterized by θ . For each query x , we obtain a set of candidate responses through policy rollout, forming a group $G_x = \{(y'_1, R_{y'_1}), \dots, (y'_n, R_{y'_n})\}$. Each $(y'_i, R_{y'_i})$ consists of a sampled response and its corresponding reward score. To normalize the advantage within each group G_x , we compute the normalized advantage $A_i(y'_i|x)$ for each response y'_i as follows:

$$A_i(y'_i|x) = \frac{R_{y'_i} - \mu_x}{\sigma_x + \xi} \quad (2)$$

where μ_x and σ_x are the mean and standard deviation of reward scores within group G_x , and ξ is a small constant added for numerical stability to avoid division by zero. The GRPO objective function $J_{\text{GRPO}}(\theta)$ is formulated to balance reward maximization and policy stability, which is defined as:

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{R_i \sim \pi_{\theta}} \left[\frac{1}{G} \sum_{i=1}^G \min \left(\frac{\pi_{\theta}(y'_i|x)}{\pi_{\text{old}}(y'_i|x)} A_i(y'_i|x), \right. \right. \\ \left. \left. \text{clip} \left(\frac{\pi_{\theta}(y'_i|x)}{\pi_{\text{old}}(y'_i|x)}, 1 - \epsilon, 1 + \epsilon \right) A_i(y'_i|x) \right) - \beta D_{\text{KL}}(\pi_{\theta} \parallel \pi_{\text{ref}}) \right] \quad (3)$$

where π_{old} is the policy before the update, ϵ is a hyperparameter controlling the clipping range, and β is a weight coefficient for the KL divergence penalty. The KL divergence term, $D_{KL}(\pi_{\theta} \parallel \pi_{\text{ref}}) = \frac{\pi_{\text{ref}}(y'_i|x)}{\pi_{\theta}(y'_i|x)} - \log \left(\frac{\pi_{\text{ref}}(y'_i|x)}{\pi_{\theta}(y'_i|x)} \right) - 1$, enforces proximity to a reference policy π_{ref} , thus preventing excessive policy shifts and mitigating the risk of over-optimization.

PASR Rollout. To enable the model to autonomously determine both whether, when and how to perform refinement during the generation process, we first design a structured output format guided by a system prompt. The prompt is shown in Figure 7.

The system prompt explicitly instructs the model to format its output using three specialized tags: `<think>`, `<refine>` and `<answer>`, which denote the reasoning trajectory, the refinement segments, and the final response, respectively. The `<think>` tag encapsulates the entire reasoning trajectory. Within this reasoning scope, the `<refine>` tag identifies specific segments where the model is expected to revise and improve previously generated content. Importantly, the `<refine>` tag required to be nested within the `<think>` tag, indicating that refinement is an integral part of the model’s reasoning process. After each `<refine>` segment, the model continues its reasoning based on the updated content, allowing refinements to directly influence subsequent inference steps. The model is encouraged to perform recursive refinement, allowing it to invoke the `<refine>` action multiple times within a single generation when it deems such actions beneficial for improving output quality.

The introduction of these special tags imposes a **semantically structured format** on the generation process, guiding the model to focus on each phase of generation, including reasoning, refinement, and final response, with explicit functional roles. The refinement answer format of PASR is shown in Figure 2.

2.3 Reward Design

Rule-based reward mechanisms have demonstrated strong empirical performance and are widely adopted in RL settings [33][34]. In our training framework, we employ a hybrid reward scheme that incorporate both rule-based and model evaluation mechanisms to guide the model’s generation and refinement behavior. Specifically, we define three types of rewards: a format reward r_{format} , an accuracy reward r_{acc} and a refinement reward r_{refine} .

Format Reward. This reward evaluates whether the generated output adheres to predefined structural constraints (as illustrated in Figure 2). The constraints are formally specified as follows:

Constraint 1 (C1): the output must include both `<think>` and `<answer>` tag pairs; the `<refine>` tag is optional.

Constraint 2 (C2): if the `<refine>` tag appears, it must be properly nested within the `<think>` tag.

Constraint 3 (C3): the relative order of the three tags must be preserved and cannot be rearranged.

Let $C_i(y' \in 0, 1)$ be a Boolean function indicating whether condition C_i is satisfied for a given output y' . The format reward $r_{\text{format}}(y')$ is then defined as:

$$r_{\text{format}}(y') = 2(C_1(y') C_2(y') C_3(y')) - 1 \quad (4)$$

This formulation assigns a reward of 1 if and only if all structural constraints are satisfied; otherwise, a penalty of -1 is applied. This strict binary scheme ensures that only fully well-formed outputs are positively reinforced.

Accuracy Reward It is designed to evaluate the quality and correctness of PASR’s generated answers. As our training tasks are drawn from open-domain question, many of which are inherently ambiguous or under-specified. Consequently, the model’s outputs are often diverse and expressed in free-form language, making evaluation methods, such as rule-based checks or exact string matching, ineffective.

To address this issue, we follow the method used in prior work [35] and employ another advanced LLM as the judge model. The evaluation model is prompted with three components: the original question x , the generated answer y' and a oracle answer \hat{y} . The judge model then outputs a continuous score in the range $[0, 1]$, reflecting the semantic quality and task relevance of the generated response relative to the reference. Let \mathcal{J} denote the judgment function instantiated by the LLM evaluator, then the accuracy reward $r_{\text{acc}}(y')$ is defined as:

$$r_{\text{acc}}(y') = \mathcal{J}(x, \hat{y}, y') \quad (5)$$

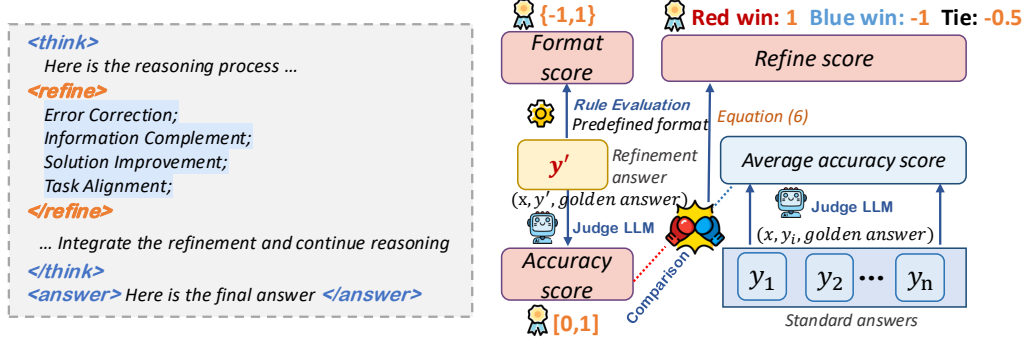


Figure 2: Answer format used in PASR (Left). Reward design for a generated answer y' during training (Right). The total reward is computed as the sum of the format score, accuracy score, and refinement score, as defined in Equation 7.

Refinement Reward. It is used to assess whether refinement actions of y' are beneficial and timely. Since directly measuring the effectiveness of adaptive self-refinement remains challenging, we instead employ a proxy evaluation strategy that assesses refinement quality by **comparing** the refined response y' with a set of standard responses y without refinement. Given the stochastic nature of the model’s generation, we sample multiple standard responses to estimate the expected accuracy of the model, denoted as $\bar{r}_{acc}(y)$. The refinement reward is designed according to the follows principles:

Reward effective refinements. A positive reward is given when the refined response achieves significantly higher accuracy than the average of standard responses.

Penalize harmful refinements. A negative reward is assigned if the refinement results in lower accuracy than the baseline average.

Discourage unnecessary refinements. When the refined response yields comparable accuracy to the average, a small penalty is applied to discourage redundant changes. Specifically, the refinement reward is then defined as:

$$r_{refine}(y') = \begin{cases} 1, & r_{acc}(y') > \bar{r}_{acc}(y) + \zeta \\ -1, & r_{acc}(y') < \bar{r}_{acc}(y) - \zeta \\ -0.5, & |r_{acc}(y') - \bar{r}_{acc}(y)| \leq \zeta \end{cases} \quad (6)$$

Here, ζ is the tolerance parameter that provides robustness against noise and minor fluctuations. This formulation encourages the model to refine its output only when the refinement yields a measurable gain, while penalizing ineffective or unnecessary modifications.

Overall Reward. The final reward for each response generated by π_θ is computed as the sum of the three components.

$$R_{y'} = r_{format}(y') + r_{acc}(y') + r_{refine}(y') \quad (7)$$

Unlike prior approaches that rely solely on binary reward signals, our **fine-grained** reward is designed to encourage meaningful and constructive refinement while explicitly discouraging both excessive and insufficient refinement.

3 Experiments

3.1 Setup

Benchmarks and Metrics. We evaluate generalization of PASR across ten datasets covering diverse tasks. For general knowledge evaluation, we use MMLU [36]. DROP [37] is included to assess multi-hop and comprehensive reasoning. Mathematical reasoning is evaluated using GSM8K [38], MATH [39], and AIME24². To test complex reasoning abilities, we adapt ARC³ and GPQA⁴. Winogrande

²<https://huggingface.co/datasets/math-ai/aime24>

³https://huggingface.co/datasets/allenai/ai2_arc

⁴<https://huggingface.co/datasets/Idavidrein/gpqa>

(Wino) [40] and CommonsenseQA (CSQA) [41] are used for knowledge-based reasoning. For summarization, we use XSum dataset ⁵. Accuracy is used as the evaluation metric for all datasets except XSum, for which we report similarity scores.

Baselines. We use Qwen2.5-7B [42] and Qwen3-8B⁶ as the backbone models, and compare PASR against several existing methods designed to induce self-improvement or self-correction abilities in LLMs. The baselines include: (1) **Self-refine** [43]: Prompts a base model to critique and iteratively revise its own responses in a single-turn format. (2) **Self-refine⁺ (with oracle)** [6]: An enhanced version of Self-Refine, where the model leverages ground truth answers to identify and revise errors after generating an initial response. (3) **PTR** [14]: Constructs a progressive self-refinement dataset and applies instruction tuning to enable multi-turn, answer-level refinement. (4) **SCoRe** [28]: Employs a multi-turn reinforcement learning framework to train LLMs to self-correct without relying on oracle feedback. (5) **STaR** [44]: Uses few-shot prompting to generate rationales for multiple questions. If the answer is incorrect, the rationale is regenerated using the correct answer. The model is iteratively fine-tuned on rationales that lead to correct outcomes. (6) **ISC** [45]: Builds a self-correction dataset and applies instruction tuning to train the model’s intrinsic self-correction ability to detect and amend its own errors. (7) **RISE** [46]: Creates improvement trajectories showing how a model can refine its own responses under its own distribution, and fine-tunes the model on these recursive rollouts. Detailed descriptions of the prompts, important parameters and implementation settings for all baselines are shown in the Appendix B.

3.2 Main Results

Unlike prior approaches that perform refinement only after the generation is complete, PASR refines answers adaptively during the generation process. To evaluate its effectiveness, we conduct experiments across a diverse set of tasks, with a focus on generalization capability. For fair comparison, we re-implement representative baselines that are only trained on specific domains under the same training data. The results are shown in Table 1.

PASR consistently outperforms baseline models, with particularly notable gains on more challenging tasks. For example, on the Qwen2.5-7B model evaluated with the MATH dataset, PASR yields a 5.2% improvement in accuracy compared to the standard method. Similarly, on the Qwen3-8B model tested with the Drop dataset, PASR achieves a 14.1% accuracy gain over the standard method. These results suggest that PASR, is capable of dynamically detecting and correcting reasoning errors, leading to effective and domain-agnostic performance gains.

PASR achieves high performance without relying on external feedback or task-specific supervision. Our experiments show that *Self-refine*, without any oracle hint from the environment or human feedback, it leads to a degradation in performance across all models. Only when oracle feedback is available to assist refinement, the self-refine⁺ provides the performance boost. This highlights the limitation of the self-refine structure in effectively improving model performance without external guidance, which is also observed in [28][46]. However, external supervision signals are often difficult to obtain and introduce additional costs. In contrast, PASR performs self-refinement autonomously, relying solely on intrinsic, self-adaptive decisions made during the generation process.

PASR demonstrates strong generalization capabilities. PASR is trained on general tasks and evaluated on domain-specific datasets to assess its generalization ability. Despite this domain shift, PASR achieves the best average performance compared to other self-refinement methods. While PASR does not always outperform all baselines on every individual dataset. For instance, its performance on Qwen2.5-7B is slightly lower on certain domain-specific tasks. This outcome is expected and understandable. Domain-specific tasks often require specialized knowledge or exhibit distributional characteristics not present in the training data. Moreover, we observe that the effectiveness of PASR can also vary with the underlying model. Compared to the more advanced Qwen3-8B, Qwen2.5-7B appears to exhibit a relatively weaker ability to leverage the learned proactive self-refinement mechanism. This suggests that stronger base models provide are fundamental to proactive self-refinement capability.

3.3 Efficiency Analysis of PASR

PASR optimizes the output quality with minimal additional token overhead. We compare token consumption across different baselines, as illustrated in Figure 3. Compared to standard decoding

⁵<https://huggingface.co/datasets/EdinburghNLP/xsum>

⁶<https://huggingface.co/Qwen/Qwen3-8B>

Table 1: PASR vs. other baselines. Compared to the base model, PASR achieves an average performance improvement of 4.8% and 8.2% on the two models, respectively.

Methods	Public	Math			Reasoning		Knowledge		Comp.	Gene.	Sum.	Avg
		GSM8K	MATH	AIME24	ARC	GPQA	Wino	CSQA	Drop	MMLU	Xsum	
Qwen2.5-7B												
Vanilla	-	88.8	68.4	16.7	85.3	25.6	64.7	62.8	78.6	46.0	31.6	56.9
Self-Refine ⁺ [6]	NIPS'23	89.6	69.4	16.7	89.0	27.7	73.8	67.5	80.2	63.0	56.2	63.3
Self-Refine[43]	NIPS'23	<u>88.7</u>	68.4	16.7	85.3	25.6	64.1	62.3	<u>78.6</u>	49.0	36.0	57.5
PTR[14]	ICLR'25	88.6	61.8	10.0	91.0	27.7	59.0	75.3	75.7	74.0	50.4	61.6
SCoRe[28]	ICLR'25	82.4	63.2	3.3	67.2	14.5	48.1	46.4	65.8	56.0	35.0	48.2
STaR[44]	NIPS'22	83.5	<u>70.8</u>	10.0	<u>88.3</u>	19.3	53.7	19.4	72.2	47.0	32.9	49.7
ISC[45]	AAAI'24	56.2	56.6	6.7	67.6	19.4	56.3	50.1	57.8	35.0	31.5	43.7
RISE[46]	NIPS'24	84.9	62.4	<u>13.3</u>	82.9	23.7	60.9	<u>74.5</u>	73.1	45.0	56.6	57.7
PASR(+prompt)	-	79.0	54.4	6.7	46.8	22.5	34.8	30.3	70.6	34.0	23.1	40.2
PASR(+IFT)	-	89.2	<u>70.8</u>	3.3	84.6	23.6	<u>62.4</u>	65.4	77.3	51.0	42.0	57.0
PASR[†]	-	88.8	73.6	10.0	86.6	29.3	57.0	67.0	79.6	75.0	49.9	61.7
Qwen3-8B												
Vanilla	-	91.3	80.2	13.3	89.0	25.0	64.5	66.3	71.2	72.0	36.3	60.9
Self-Refine ⁺ [6]	NIPS'23	94.8	84.4	23.3	94.0	43.7	83.0	83.5	85.0	85.0	51.1	72.8
Self-Refine[43]	NIPS'23	90.5	73.0	10.0	<u>91.3</u>	29.1	<u>76.8</u>	<u>75.8</u>	80.8	<u>73.0</u>	<u>50.2</u>	<u>65.0</u>
PTR[14]	ICLR'25	88.7	72.0	6.7	80.9	32.3	66.1	46.4	65.5	53.0	33.7	54.5
SCoRe[28]	ICLR'25	91.4	<u>81.2</u>	<u>13.3</u>	87.3	36.7	70.7	63.9	78.9	72.0	45.0	64.0
STaR[44]	NIPS'22	72.7	55.2	0.0	64.2	26.0	55.3	28.8	49.5	22.0	13.7	38.7
ISC[45]	AAAI'24	23.6	57.2	6.7	68.2	29.2	63.5	28.3	42.5	28.0	38.3	38.6
RISE[46]	NIPS'24	92.5	77.4	16.7	88.3	33.3	70.8	37.2	<u>82.4</u>	44.0	49.3	59.2
PASR(+prompt)	-	60.3	67.8	10.0	57.9	29.4	60.4	74.3	75.1	52.0	26.6	51.4
PASR(+IFT)	-	<u>91.7</u>	74.6	6.7	73.6	<u>35.1</u>	68.7	29.3	73.5	36.0	36.3	52.6
PASR[†]	-	94.9	81.4	16.7	92.3	24.5	80.0	79.6	85.3	83.0	53.0	69.1

method, PASR achieves notable accuracy gains with only a slight increase in token usage. This highlights its ability to enhance outputs through targeted, dynamic refinements rather than full rewrites, making it a cost-efficient refinement method. Specifically, on the Qwen2.5-7B, PASR yields a 4.8% absolute performance improvement with only an 8.4% increase in token consumption compared to standard generation.

Additionally, while PASR and PTR achieve comparable performance on Qwen2.5-7B, PTR incurs significantly higher token costs. The performance gain of PTR mainly stems from the use of high-quality, answer-level refinement data. However, the effectiveness of this data diminishes considerably on Qwen3-8B. However, PTR regenerates entire answers at each refinement step, resulting in substantial token overhead.

3.4 In-depth Analysis

3.4.1 Does PASR genuinely exhibit proactive refinement capabilities during generation?

To further verify the effectiveness of PASR's in-process refinement, we conduct a quantitative analysis from three aspects, including refinement ratio, refinement coherence and the causal relationship between refinement trajectories and final answers. Prompts used in this subsection are shown in Figure 12 and 13 The results are shown in the Figure 4.

PASR autonomously determine when to refine. We randomly sample 384 questions, among which 267 are initially answered incorrectly by the base model. PASR does not refine all answers indiscriminately; instead, it selectively triggers refinement. Among the 267 incorrect answers, 235 are revised and corrected by PASR. While many originally correct answers nearly remain unchanged. This indicates that PASR is able to identify and act upon potentially flawed generations when refinement is necessary.

PASR shows high coherence between pre- and post-refinement outputs. We randomly sample 300 answers and employ an independent LLM, Qwen2.5-32B-Instruct, to evaluate their semantic consistency before and after refinement. Each sample is scored multiple times within in $[0, 1]$ to ensure the reliability of the assessment. The results indicate that nearly 80% of samples received a semantic consistency score exceeding 0.9.

PASR's proactive self-refinement process contributes to the answer correctness. We further analyze the 300 samples mentioned above to evaluate the alignment between the refinement process

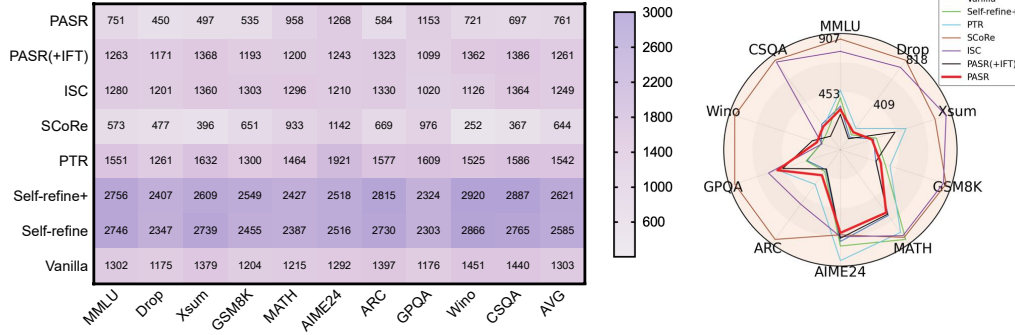


Figure 3: Token usage comparison of different methods across various tasks. The left figure shows results based on the Qwen3-8B backbone, while the right figure based on the Qwen2.5-7B.

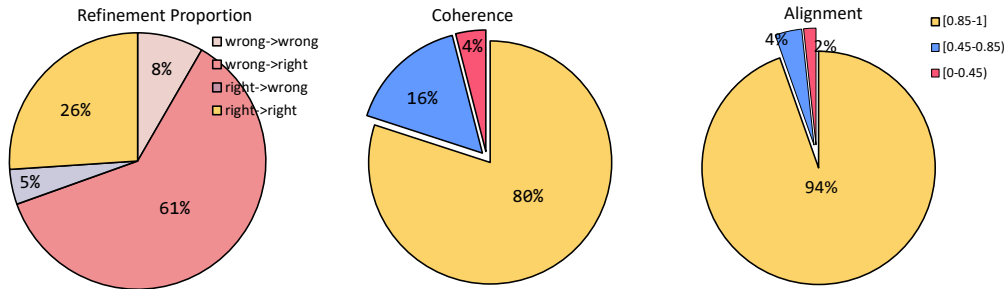


Figure 4: From left to right, the pie charts show: (1) the proportion of answers changed by PASR refinement, (2) the distribution of coherence scores reflecting how well the self-refinement builds upon the initial generation, and, and (3) the distribution of alignment scores measuring the consistency between the refinement process and the final answer. For (2) and (3), each segment represents the proportion of examples falling within a specific score range (e.g., [0–0.45], [0.45–0.85], [0.85–1.0]).

276 and the final answer. Over 85% of the samples achieved a alignment score above 0.9, indicating that
277 refinement leads to the quality of outputs.

278 3.4.2 What makes PASR effective?

279 **Reinforcement learning enables the model to perform proactive self-refinement.** In contrast,
280 *prompt-based or supervised signals are insufficient to elicit proactive refinement capabilities.* We
281 explore whether proactive self-refinement can be induced via prompting. The results are shown in
282 Table 1. When the model is explicitly instructed to self-refine during generation via prompt design
283 (PASR+prompt), we observe a *consistent performance decline* across all tasks, with an average
284 decrease of 16.9% and 9.5% on two backbone models. It indicates that prompt-based guidance alone
285 is insufficient to elicit the model’s adaptive self-refinement capability.

286 Similarly, we apply instruction-following finetuning (PASR+IFT) to inject this capability. However,
287 the model shows *limited generalization* to unseen tasks. On the Qwen3-8B model, performance drops
288 by 8.3% compared to the base version. These results suggest that proactive self-refinement is not an
289 innate capability and cannot be effectively acquired through supervised fine-tuning.

290 **Comparison-based rewards setting help the model learn to perform effective refinements.** We
291 use Qwen2.5-7B as the backbone and evaluate two alternative reward strategies. The first is *Single-*
292 *reference comparison (w/o multi-answer)*, computes refinement rewards by comparing the refined
293 output to a *single standard answer*. The second is *Refinement-triggered reward (w/o comparison)*,
294 assigns a coarse positive refinement reward whenever a refinement action is taken, regardless of its
295 necessity or effectiveness. The results are shown in Table 2.

296 Although both alternative strategies show moderate performance, they consistently under perform
297 compared to our proposed reward design. Our method computes the refinement reward by comparing
298 the refined output to the average score across multiple standard answers, providing a more stable and
299 reliable evaluation. This reward strategy offers several key advantages.

Table 2: PASR performance across datasets under different refinement reward signals. The comparison-based fine-grained reward better guides the model to learn adaptive and meaningful refinements.

Dataset	PASR	w/o multi-answer	w/o comparison
MMLU	75.0	71.0 (-4.0)	53.0 (-22.0)
Drop	79.6	76.7 (-2.9)	78.6 (-1.0)
Xsum	49.9	44.3 (-5.6)	31.9 (-18.0)
GSM8K	88.8	75.7 (-13.1)	86.0 (-2.8)
MATH	73.6	62.2 (-11.4)	62.2 (-11.4)
AIME24	10.0	10.0 (+0.0)	10.0 (+0.0)
ARC	86.6	83.9 (-2.7)	82.9 (-3.7)
GPQA	29.3	28.9 (-0.4)	27.4 (-1.9)
Wino	57.0	53.4 (-3.6)	65.3 (+8.3)
CSQA	67.0	65.9 (-1.1)	64.9 (-2.1)
AVG	61.7	57.2 (-4.5)	56.2 (-5.5)

First, averaging over multiple standard answers reduces the variance introduced by the randomness of LLM outputs. It provides a more *robust and stable* learning signal for guiding meaningful refinements during training. This strategy enables the model to better recognize when a refinement yields a genuine improvement. Moreover, coarse-grained reward signals are easily exploited by the model, leading to unnecessary refinement in pursuit of high reward (i.e., *reward hacking*). In contrast, our comparison-based signal avoids this by rewarding only measurable improvements, leading to more targeted and meaningful refinements.

4 Related Work

prompt-based self-refinement. Prior work on self-refinement typically follows a two-stage paradigm. The model first generates an initial response and is then prompted to refine or improve it [10]. These methods have seen widespread use in complex reasoning tasks, including math [47][48] and code generation [49][11][12]. However, simply prompting a model to refine its own output does not consistently yield better results, and there is little evidence that prompting alone is sufficient for reliable self-improvement[8][50]. Success in these settings often relies on the availability of ground truth feedback or external supervision, such as explicit information about the error, its location, and an explanation of why it is wrong [51][43]. Unfortunately, such fine-grained feedback is rarely accessible in practical applications [17][52]. Therefore, some studies utilize stronger models or train auxiliary teacher models to evaluate outputs and provide feedback [4][6][53][7]. While effective, these approaches usually require task-specific annotations to train the feedback models, which significantly increases the cost and limits scalability across diverse tasks [14].

Fine-tuning for self-refinement. Another line of work focuses on supervised fine-tuning using synthetic self-refinement data. In these settings, initial answers are generated by one model, while refined answers are produced by a stronger model or taken from oracle answers [13][14][45] [4]. The resulting pairs of “bad” to “good” answers are used to train models to imitate the refinement process. However, such methods suffer from either distributional mismatch, where the errors in training data do not reflect the mistakes the model makes during inference [54], or behavioral collapse, where the model learns a narrow correction pattern that fails to generalize across tasks or domains [28][46].

5 Conclusion

We propose PASR, a novel method that enables large language models to proactively self-refine their responses during generation. PASR leverages an on-policy reinforcement learning approach to explore whether, when, and how to perform refinements. We design fine-grained rewards to encourage effective refinements and penalize incorrect or unnecessary ones. Experiments show that PASR achieves a strong balance between performance and efficiency. Moreover, even when trained only on general open-domain data, PASR achieves strong self-refinement across ten diverse tasks, demonstrating strong generalization not observed in previous work.

References

- [1] John Dewey. Experience and education. In *The educational forum*, volume 50, pages 241–252. Taylor & Francis, 1986.
- [2] Julius Kuhl and Jürgen Beckmann. *Action control: From cognition to behavior*. Springer Science & Business Media, 2012.
- [3] Yongqi Tong, Dawei Li, Sizhe Wang, Yujia Wang, Fei Teng, and Jingbo Shang. Can LLMs learn from previous mistakes? investigating LLMs’ errors to boost for reasoning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3065–3080, Bangkok, Thailand, August 2024. Association for Computational Linguistics.
- [4] Zhihui Xie, Jie chen, Liyu Chen, Weichao Mao, Jingjing Xu, and Lingpeng Kong. Teaching language models to critique via reinforcement learning. In *ICLR 2025 Third Workshop on Deep Learning for Code*, 2025.
- [5] Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. Learning from mistakes makes llm better reasoner, 2024.
- [6] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhunoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: iterative refinement with self-feedback. In *Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS ’23*, Red Hook, NY, USA, 2023. Curran Associates Inc.
- [7] Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. Generating sequences by learning to self-correct. In *The Eleventh International Conference on Learning Representations*, 2023.
- [8] 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*, 2024.
- [9] Deep Ganguli, Amanda Askell, Nicholas Schiefer, Thomas I. Liao, Kamilè Lukošiušė, Anna Chen, Anna Goldie, Azalia Mirhoseini, Catherine Olsson, Danny Hernandez, Dawn Drain, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jackson Kernion, Jamie Kerr, Jared Mueller, Joshua Landau, Kamal Ndousse, Karina Nguyen, Liane Lovitt, Michael Sellitto, Nelson Elhage, Noemi Mercado, Nova DasSarma, Oliver Rausch, Robert Lasenby, Robin Larson, Sam Ringer, Sandipan Kundu, Saurav Kadavath, Scott Johnston, Shauna Kravec, Sheer El Showk, Tamara Lanham, Timothy Telleen-Lawton, Tom Henighan, Tristan Hume, Yuntao Bai, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, Christopher Olah, Jack Clark, Samuel R. Bowman, and Jared Kaplan. The capacity for moral self-correction in large language models, 2023.
- [10] Deep Ganguli, Amanda Askell, Nicholas Schiefer, Thomas I. Liao, Kamilè Lukošiušė, Anna Chen, Anna Goldie, Azalia Mirhoseini, Catherine Olsson, Danny Hernandez, Dawn Drain, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jackson Kernion, Jamie Kerr, Jared Mueller, Joshua Landau, Kamal Ndousse, Karina Nguyen, Liane Lovitt, Michael Sellitto, Nelson Elhage, Noemi Mercado, Nova DasSarma, Oliver Rausch, Robert Lasenby, Robin Larson, Sam Ringer, Sandipan Kundu, Saurav Kadavath, Scott Johnston, Shauna Kravec, Sheer El Showk, Tamara Lanham, Timothy Telleen-Lawton, Tom Henighan, Tristan Hume, Yuntao Bai, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, Christopher Olah, Jack Clark, Samuel R. Bowman, and Jared Kaplan. The capacity for moral self-correction in large language models, 2023.
- [11] Theo X. Olausson, Jeevana Priya Inala, Chenglong Wang, Jianfeng Gao, and Armando Solar-Lezama. Is self-repair a silver bullet for code generation? In *The Twelfth International Conference on Learning Representations*, 2024.

- [12] Theo X. Olausson, Jeevana Priya Inala, Chenglong Wang, Jianfeng Gao, and Armando Solar-Lezama. Demystifying gpt self-repair for code generation. *CoRR*, abs/2306.09896, 2023.
- [13] Alex Havrilla, Sharath Rapparthi, Christoforos Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravynski, Eric Hambro, and Roberta Raileanu. Glore: when, where, and how to improve llm reasoning via global and local refinements. In *Proceedings of the 41st International Conference on Machine Learning*, ICML’24. JMLR.org, 2024.
- [14] Chengyu Du, Jinyi Han, Yizhou Ying, Aili Chen, Qianyu He, Haokun Zhao, Sirui Xia, Haoran Guo, Jiaqing Liang, Zulong Chen, et al. Think thrice before you act: Progressive thought refinement in large language models. In *The Twelfth International Conference on Learning Representations*, 2025.
- [15] Zeyu Gan, Yun Liao, and Yong Liu. Rethinking external slow-thinking: From snowball errors to probability of correct reasoning, 2025.
- [16] Gregor Bachmann and Vaishnavh Nagarajan. The pitfalls of next-token prediction, 2024.
- [17] 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.
- [18] 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.
- [19] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-rl: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- [20] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2998–3009, 2023.
- [21] Gautier Dagan, Frank Keller, and Alex Lascarides. Dynamic planning with a llm. *arXiv preprint arXiv:2308.06391*, 2023.
- [22] Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*, 2024.
- [23] Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, et al. From generation to judgment: Opportunities and challenges of llm-as-a-judge. *arXiv preprint arXiv:2411.16594*, 2024.
- [24] Zichen Liu, Changyu Chen, Wenjun Li, Tianyu Pang, Chao Du, and Min Lin. There may not be aha moment in rl-zero-like training — a pilot study. <https://oatllm.notion.site/oat-zero>, 2025. Notion Blog.
- [25] Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, and Shiguo Lian. Dast: Difficulty-adaptive slow-thinking for large reasoning models. *arXiv preprint arXiv:2503.04472*, 2025.
- [26] Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for $2+3=?$ on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024.
- [27] Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, et al. Thoughts are all over the place: On the underthinking of o1-like llms. *arXiv preprint arXiv:2501.18585*, 2025.

- [28] Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2025.
- [29] Yubo Wang, Xiang Yue, and Wenhua Chen. Critique fine-tuning: Learning to critique is more effective than learning to imitate, 2025.
- [30] Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Ren Lu, Thomas Mesnard, Johan Ferret, Colton Bishop, Ethan Hall, Victor Carbune, and Abhinav Rastogi. RLAIIF: Scaling reinforcement learning from human feedback with AI feedback, 2024.
- [31] Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. Self-rewarding language models. In *Forty-first International Conference on Machine Learning*, 2024.
- [32] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.
- [33] Alan Dao and Dinh Bach Vu. Alphamaze: Enhancing large language models’ spatial intelligence via grpo. *arXiv preprint arXiv:2502.14669*, 2025.
- [34] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024.
- [35] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.
- [36] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.
- [37] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In Jill Burstein, Christy Doran, and Tamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.
- [38] 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, 2021.
- [39] 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.
- [40] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, August 2021.
- [41] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In Jill Burstein, Christy Doran, and Tamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.
- [42] 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, Jiayi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu,

- 481 Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji
482 Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang
483 Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5
484 technical report, 2025.
- 485 [43] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao.
486 Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information*
487 *Processing Systems*, 36:8634–8652, 2023.
- 488 [44] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STar: Bootstrapping reasoning with
489 reasoning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors,
490 *Advances in Neural Information Processing Systems*, 2022.
- 491 [45] Haixia Han, Jiaqing Liang, Jie Shi, Qianyu He, and Yanghua Xiao. Small language model can
492 self-correct. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages
493 18162–18170, 2024.
- 494 [46] Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. Recursive introspection: Teaching
495 language model agents how to self-improve. In *The Thirty-eighth Annual Conference on Neural*
496 *Information Processing Systems*, 2024.
- 497 [47] Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and
498 Jun Zhao. Large language models are better reasoners with self-verification. In Houda Bouamor,
499 Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics:*
500 *EMNLP 2023*, pages 2550–2575, Singapore, December 2023. Association for Computational
501 Linguistics.
- 502 [48] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu,
503 and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human
504 annotations. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of*
505 *the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long*
506 *Papers)*, pages 9426–9439, Bangkok, Thailand, August 2024.
- 507 [49] Theo X. Olausson, Jeevana Priya Inala, Chenglong Wang, Jianfeng Gao, and Armando Solar-
508 Lezama. Demystifying gpt self-repair for code generation. *CoRR*, abs/2306.09896, 2023.
- 509 [50] Gladys Tyen, Hassan Mansoor, Victor Carbune, Peter Chen, and Tony Mak. LLMs cannot find
510 reasoning errors, but can correct them given the error location. In Lun-Wei Ku, Andre Martins,
511 and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL*
512 *2024*, pages 13894–13908, Bangkok, Thailand, August 2024. Association for Computational
513 Linguistics.
- 514 [51] Geunwoo Kim, Pierre Baldi, and Stephen McAleer. Language models can solve computer tasks.
515 In *Proceedings of the 37th International Conference on Neural Information Processing Systems*,
516 *NIPS ’23*, Red Hook, NY, USA, 2023. Curran Associates Inc.
- 517 [52] Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang
518 Wang. Automatically correcting large language models: Surveying the landscape of diverse
519 automated correction strategies. *Transactions of the Association for Computational Linguistics*,
520 12:484–506, 2024.
- 521 [53] Jonathan Uesato, Nate Kushman, Ramana Kumar, H. Francis Song, Noah Yamamoto Siegel,
522 Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems
523 with process-based and outcome-based feedback, 2023.
- 524 [54] Katie Kang, Eric Wallace, Claire Tomlin, Aviral Kumar, and Sergey Levine. Unfamiliar
525 finetuning examples control how language models hallucinate. In Luis Chiruzzo, Alan Ritter, and
526 Lu Wang, editors, *Proceedings of the 2025 Conference of the Nations of the Americas Chapter*
527 *of the Association for Computational Linguistics: Human Language Technologies (Volume*
528 *1: Long Papers)*, pages 3600–3612, Albuquerque, New Mexico, April 2025. Association for
529 Computational Linguistics.

A Limitations

Despite the promising results achieved by PASR, there still are some limitation remain. Similar to existing self-refinement methods, the performance evaluation of PASR primarily focuses on the quality of the final answers. Although we analyze PASR’s refinement trajectory from three aspects to demonstrate its ability to perform genuine refinement (as shown in Section 3.4), only the change in answer correctness before and after refinement serves as the objective metric. The coherence is evaluated with the assistance of a large language model, introducing a degree of subjectivity. However, given the free-form nature of text generation, we have not yet established objective and reliable metrics to deeply and systematically analyze the refinement trajectory. In future work, we aim to design more rigorous and scientifically grounded metrics to better quantify the effectiveness of the refinement trajectory during the generation process.

B Experimental Details

Platform. All of our experiments are conducted on workstations equipped with eight NVIDIA A800 PCIe GPUs with 80GB memory, running Ubuntu 20.04.6 LTS and PyTorch 2.0.1.

Important Parameters of PASR. The PASR is implemented based on the open-source GitHub repository⁷. The KL divergence penalty coefficient β is set to 0.04 to balance policy improvement and deviation from the reference policy. The clipping parameter ϵ is set to 0.2. For each group, 8 answers are generated, and the training batch size is set to 2.

Distributed training utilizes the DeepSpeed library with the *AdamW* optimizer and a learning rate of $1e-6$. Gradient accumulation occurs over 4 steps, and with a per-GPU batch size of 2, the effective batch size is $8 \times N_{\text{GPUs}}$, where N_{GPUs} denotes the number of GPUs.

Mixed-precision training with BF16 is enabled. Memory optimization employs ZeRO Stage 2, with optimizer state offloading to CPU. Key ZeRO configurations include allgather partitions, an allgather bucket size of 2e8, reduce scatter, and a reduce bucket size of 2e8. Contiguous gradients are enabled, communication overlap is disabled, and 16-bit weights are gathered during model saving. Training loss is logged every 5 steps.

B.1 Implementation Details for Baselines

We use the LLaMA-Factory framework⁸ to train all baseline methods. The key parameters are shown in the Table 3.

⁷https://github.com/lsdefine/simple_GRP0

⁸<https://github.com/hiyouga/LLaMA-Factory>

Table 3: Important parameters for each baseline method

Method	Parameters
PTR	per_device_train_batch_size: 1 gradient_accumulation_steps: 2 learning_rate: 1.0×10^{-5} num_train_epochs: 2 lr_scheduler_type: cosine warmup_ratio: 0.1 bf16: true Dataset: Public GitHub
SCoRe	per_device_train_batch_size: 1 gradient_accumulation_steps: 4 learning_rate: 1.0×10^{-5} num_train_epochs: 2.0 lr_scheduler_type: cosine warmup_ratio: 0.1 bf16: true Dataset: preference pairs form PTR experiment
STaR	per_device_train_batch_size: 1 gradient_accumulation_steps: 2 learning_rate: 1.0×10^{-5} num_train_epochs: 2 lr_scheduler_type: cosine warmup_ratio: 0.1 bf16: true Dataset: alpaca_evol_instruct_70k(filtered generated pairs))
ISC	per_device_train_batch_size: 1 gradient_accumulation_steps: 2 learning_rate: 1.0×10^{-5} num_train_epochs: 2.0 lr_scheduler_type: cosine warmup_ratio: 0.1 bf16: true Dataset: alpaca_evol_instruct_70k
RISE	per_device_train_batch_size: 1 gradient_accumulation_steps: 2 learning_rate: 1.0×10^{-5} num_train_epochs: 2.0 lr_scheduler_type: cosine warmup_ratio: 0.1 bf16: True Dataset: alpaca_evol_instruct_70k
PASR(+IFT)	per_device_train_batch_size: 1 gradient_accumulation_steps: 2 learning_rate: 1.0×10^{-5} num_train_epochs: 2.0 lr_scheduler_type: cosine warmup_ratio: 0.1 bf16: True Dataset: good refinement paths generated during PASR training

Question:

A regular hexagon can be divided into six equilateral triangles. If the perimeter of one of the triangles is 21 inches, what is the perimeter, in inches, of the regular hexagon?

Answer:

<think>Okay, let's see. The problem says a regular hexagon can be divided into six equilateral triangles.

...

So the perimeter of the hexagon should be 42 inches. Wait, but let me double-check.

<refine>If each triangle has sides of 7 inches, then the sides of the hexagon are 7 inches each. So six sides would be $6 \times 7 = 42$. </refine>That seems right. But maybe I'm missing something here. Let me think again. <refine>The perimeter of the triangle is 21, so each side is 7. The triangles are equilateral, so all sides are 7. The hexagon's side is equal to the triangle's side. Therefore, the hexagon's perimeter is $6 \times 7 = 42$. Yeah, that seems correct. I don't think I need to refine anything here. The answer should be 42 inches.</refine>

</think><answer>The perimeter of the regular hexagon is 42 inches.</answer>

Figure 5: An example of PASR: the model analyzes the problem from multiple perspectives post-result acquisition to ensure solution accuracy.

Question:

What must a student in engineering do?

A: answer question B: learn language C: do mathematics

D: be able to count E: begin to study

Answer:

<think>Okay, let's tackle this question. The user is asking what a student in engineering must do, with options A to E. First, I need to understand each option.

...

the answer is C: do mathematics. <refine>Let me reconsider the options. While "do mathematics" (C) is essential, the question asks what a student in engineering "must" do. The phrase "begin to study" (E) is more universally applicable to all students, but it's not specific to engineering. However, the question might be testing the understanding that engineering students must start their academic journey, which includes studying. Therefore, "begin to study" (E) could be the most comprehensive answer, as it encompasses the foundational step of starting education, which includes all other activities like answering questions, learning languages, doing mathematics, and counting. Hence, the answer is E. </refine>

</think><answer>E: begin to study</answer>

Figure 6: An example of PASR: the model refines its initial answer by exploring alternatives and rectifying prior oversights for enhanced correctness.

559 **B.1.1 Case Study of PASR**

560 We present some examples that PASR showcase the proactive self-refinement capability. The cases
561 are shown in Figure 5 and 6

Prompt Template for PASR

System: You are a helpful assistant with self-refinement capability. After the user asks a question, you first think carefully and then give the answer.
The thinking process and answer should be enclosed within <think> </think> and <answer> </answer> tags respectively. Note that you can only use once these four tags.
In the <think> and </think> tag, follow these rules:
Start with an initial thought process on how to approach the question.
when you determine that additional clarification, detail, or improved reasoning is necessary, insert <refine> </refine> tag and then specify what needs to be reconsidered or improved.
You can use both tags multiple times.
Continue to advance your reasoning after each refinement until you feel there is no more room for improvement.
This is how your full response should be structured:
<think>Here is your thinking process, when you think you need to reflect, insert <refine>your refinement</refine>. Repeat the iterative process as many times as necessary before moving to the final answer.</think><answer>Here is an answer at the end of the thinking process.</answer>

Figure 7: Prompt template for PASR. We used it to guide the LLM to perform refinement during generation, and employed another LLM to evaluate the quality of generated outputs.

Prompt Template for PASR evaluation

You are a judge, you will judge the correctness of the answer to the question. Below is a question, a ground truth answer, and an answer generated by an AI assistant, please rate the AI assistant's answers according to the question on a scale from 0 to 1. Your output is just a number in the range from 0 to 1.
Question:
{Question}

Ground Truth:
{Ground Truth}

Answer:
{Answer}

Figure 8: Prompt template used for PASR evaluation during training. This prompt guides the judge model in evaluating the answers generated by the model during the rollout process.

B.1.2 Detailed Prompts

This section presents the detailed prompts used in our experiments, including the PASR system prompt and the evaluation prompts for different datasets, among others, as illustrated in figs. 7 to 13.

Evaluation Prompt Template for Summary Questions

Now, I want to test an AI assistant's ability to summary. Below is a text (Question), a ground truth summary (Ground Truth Answer), and an answer (Answer) generated by an AI assistant. Please rate the AI assistant's answers according to the ground truth answer. Please score answers according to how relevant they are to the text and ground truth summary. Your output is from 0 to 1, which 0 is not similar at all, 1 is basically error free.

Question:{Question} Ground Truth:{Ground Truth} Answer:{Answer}

Evaluation Prompt Template for Multiple-Choice Questions

Now, I want to test an AI assistant's ability to answer questions. Below is a multi-choice question, a ground truth answer (one of the option), and an answer generated by an AI assistant. Please rate the AI assistant's answers according to the question and the ground truth answer. If you think the answer is correct, your output is 1; otherwise, your output is 0. Your output is just 0 or 1.

Question:{Question} Ground Truth:{Ground Truth} Answer:{Answer}

Evaluation Prompt Template Open Questions

Now, I want to test an AI assistant's ability to answer questions. Below is a open question, a ground truth answer, and an answer generated by an AI assistant. Please rate the AI assistant's answers according to the ground truth answer. If you think the answer is correct, your output is 1; otherwise, your output is 0. Your output is just 0 or 1.

Question:{Question} Ground Truth:{Ground Truth} Answer:{Answer}

Figure 9: Evaluation prompt template during the test stage. We design different prompts for different types (Summary, Multi-choice and Open question) of test datasets to ensure accurate evaluation.

Prompt Template for Refinement with Oracle (Math Questions)

There might be an error in the solution above because of lack of understanding of the question. Please correct the error, if any, and rewrite the solution. Only output the final solution! At the end of the Solution, when you give your final answer, write it in the form 'Final Answer: The final answer is $\boxed{\text{answer}}$ '. I hope it is correct.

previous solution:{Initial answer}

Prompt Template for Refinement without Oracle (Open Questions)

There is an error in the previous solution. Please review each step to identify the mistake, and then provide a corrected version of the solution.

previous solution:{Initial answer}

Prompt Template for Refinement without Oracle

Please review each step of the previous solution to identify any potential errors. If you find any issues, provide a revised and corrected version of the solution. If there are no issues, simply respond with: I believe the above solution is correct.

previous solution:{Initial answer}

Figure 10: Prompt template for refinement method *self-refine* and *self-refine*⁺.

Standard Prompt for MMLU

Here is a multiple-choice question, which from a dataset tests knowledge across 57 diverse fields such as elementary mathematics, history, computer science, and law. please think step by step and give me your final answer.

Standard Prompt for Drop

Here is a passage and a question, which requires discrete reasoning over the provided text. Please think step by step and give me your final answer.

Standard Prompt for Xsum

Here is a passage. please summarize this passage.

Standard Prompt Template for Math (GSM8K, MATH, AIME24)

Here is a problem. please think step by step and give me your final answer.

Standard Prompt for ARC

Here is a multiple-choice question, which from a collection of questions for the science exam. Please think step by step and give me your final answer.

Standard Prompt for Wino

Here is a question provides two options. Please think step by step and select the correct answer based on the semantics of the sentence.

Standard Prompt for CommonsenseQA

Here is multiple-choice about commonsense. Please think step by step and give me your final answer.

Figure 11: Evaluation prompt template during the test stage. We design different prompts for MMLU, Drop, Xsum, Math type, ARC, Wino, and CommonsenseQA to ensure accurate evaluation.

Prompt for Evaluating the Reasonableness of the Refinement Process

Role

You are an AI Analyzer specializing in assessing the quality of refinement thinking.

Task

Your task is to evaluate the "reasonableness" of the refinement part within a given response. This response typically contains two parts: an initial thought or response (pre-refinement), and a part where the user reflects on that initial thought (post-refinement).

Definition of "Reasonableness"

"Reasonableness" here has a specific meaning: it measures the **coherence and consistency** between the pre-refinement and post-refinement thought processes.

You need to determine:

1. Is the refinement **based on** the preceding thought content?
2. Does the refinement process **logically follow** from the previous thinking? Or, if the refinement leads to a **shift in perspective**, is this shift explained or internally logical and understandable?
3. Does the conclusion or state after refinement form an understandable and **coherent thought trajectory** with the pre-refinement state?

Crucially: You are **not** evaluating the depth of the refinement itself, nor the correctness of the final answer. You are evaluating **only** whether the **act of refinement** is **coherent and consistent** with the preceding thought content.

Evaluation Criteria & Score

Please provide a floating-point score between **0.0** and **1.0** based on the following criteria:

- * **0.0:** Completely unreasonable. The refinement is entirely unrelated to the previous thinking, or contradicts it without any explanation. The thought process is broken or disconnected.

- * **0.5:** Partially reasonable. The refinement has some connection to the previous thinking, but the link is weak, the logical chain is unclear, or a shift in perspective seems somewhat abrupt but has a faintly traceable thread.

- * **1.0:** Highly reasonable. The refinement is clearly built upon the previous thinking, the logic is coherent, and even if perspectives shift, the reasons and process are clear, demonstrating high consistency in the thought trajectory.

Output Requirements

- * **Strictly output only a single number**, which must be a floating-point number between 0.0 and 1.0.

- * **Do not include any** explanations, justifications, text descriptions, units, or any other extra characters.

Response Text to Evaluate

Figure 12: Prompt for evaluating the reasonableness of the refinement trajectory in PASR. This prompt is used to assess whether the model-generated answers evolve in a reasonable manner throughout the refinement process.

Prompt for Evaluating the Consistency between the Refinement and the Final Answer

Role

You are an AI Analyzer specializing in evaluating thought coherence.

Task

Your task is to evaluate the consistency between a given "Thought Process" (which may include refinement) and the final "Answer".

Definition of "Consistency"

"Consistency" here measures: ****The degree to which the final answer is a direct, relevant, and logical product of the thought process.****

You need to determine:

1. Does the final answer directly address or resolve the problems, dilemmas, or goals explored in the thought process?
2. Is the final answer logically aligned with the thought process, including insights or conclusions derived from refinement?
3. Are the key information, reasoning steps, or refinements from the thought process reflected or applied in the final answer?

****Focus:**** You are ****not**** evaluating the quality of the thought process itself, nor the correctness or merit of the answer itself. You are evaluating ****only the degree of relevance and logical connection between the thought process and its final answer.****

Evaluation Criteria & Score

Please provide a floating-point score between ****0.0 and 1.0**** based on the following criteria:

- **0.0:**** Completely inconsistent/irrelevant. The final answer has little to no relation to the thought process, appears out of nowhere, or completely ignores the reasoning path.
- **0.5:**** Partially consistent/relevant. The final answer has some connection to the thought process, but might only address parts of it, the logical link might be weak, or the answer, while related, doesn't seem like the most direct conclusion from the process.
- **1.0:**** Highly consistent/relevant. The final answer clearly, directly, and logically stems from the provided thought process, serving as its definite conclusion or solution.

Output Requirements

- **Strictly output only a single number****, which must be a floating-point number between 0.0 and 1.0.
- **Do not include any**** explanations, justifications, text descriptions, units, or any other extra characters.

Response Text to Evaluate

<think> </think> is thinking process, <answer> </answer> is final answer.

Figure 13: Prompt for evaluating the alignment between the refinement process and the final answer in PASR.

C NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: Abstract and Section 1

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: Appendix

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[NA\]](#)

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: We have detailed the data, methods, and important parameters used in the experiments in both section 3 and the appendix. Furthermore, we have provided a link to an anonymous repository in the abstract.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We have detailed the data, methods, and important parameters used in the experiments in both section 3 and the appendix. Furthermore, we have provided a link to an anonymous repository in the abstract.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: These are described in detail in both the experimental section and the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: We set the temperature to 0 for all models.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: These are described in detail in both the experimental section and the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: Section 2

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: There is no societal impact of the work performed.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Section 3 and Appendix

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset’s creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: the paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

876 Question: Does the paper describe the usage of LLMs if it is an important, original, or
877 non-standard component of the core methods in this research? Note that if the LLM is used
878 only for writing, editing, or formatting purposes and does not impact the core methodology,
879 scientific rigorousness, or originality of the research, declaration is not required.

880 Answer: [Yes]

881 Justification: Section 3

882 Guidelines:

- 883 • The answer NA means that the core method development in this research does not
884 involve LLMs as any important, original, or non-standard components.
- 885 • Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>)
886 for what should or should not be described.