Test-Time Scaling with Reflective Generative Model

Anonymous ACL submission

Abstract

We introduce a new Reflective Generative Model(RGM), which obtains OpenAI o3-mini's performance via a novel Reflective Generative Form. This form focuses on high-quality reasoning trajectory selection and contains two novelties: 1) **A unified interface for policy and process reward model**: we share the backbone network and use task-specific heads for reasoning trajectory predicting and scoring respectively, introducing only 53M extra parameters for trajectory scoring. 2) **Eliminating the reliance on process-level annotation**: we provide a self-supervised process reward model, which can directly learn the high-quality reasoning trajectory selection from the outcome reward. Equipped with the reflective generative form, RGM is naturally suitable for test-time scaling, and we provide three reasoning effort modes (low, medium, and high) based on the controllable thinking length. Experiments demonstrate that our RGM achieves comparable performance to OpenAI o3-mini's series with only 32B parameter size. Code will be available.

1 Introduction

Over the past two years, the field of Large Language Models (LLMs) has experienced rapid advancements, marked by the emergence of increasingly sophisticated models. Notable developments include OpenAI's GPT-4, Google's Gemini, Meta's LLaMA series, Alibaba's Qwen, and DeepSeek's R1, which have collectively pushed the boundaries of natural language understanding and generation. This progress is attributed to innovations in model architectures and training techniques, enabling LLMs to process and generate content across various formats.

Recent analyses suggest that OpenAI's o3 model achieves its advanced reasoning and coding capabilities through Test-Time Scaling (TTS) techniques such as massive sampling, candidate scoring, and search over multiple reasoning paths (Labs, 2025; Zeff, 2024). For instance, during ARC-AGI and competitive coding evaluations, o3 was shown to generate up to 1024 candidate samples for each query (Chollet, 2024; OpenAI, 2025). These inference-time strategies mark a significant shift from traditional one-pass models, enabling o3 to adapt dynamically to novel tasks and achieve near-human performance in reasoning benchmarks.

TTS approaches can be categorized into two types: internal TTS and external TTS. Internal TTS (also called sequential TTS in Zeng et al. (2025)) strategies use CoT for longer thinking processes (Guo et al., 2025; OpenAI, 2024), which benefits from Long-CoT Supervised Fine-Tuning and reinforcement learning. Recent internal TTS methods (Guo et al., 2025) mainly suffer from the false positive reasoning process, as the outcome reward will misclassify the correct answer with incorrect reasoning during the training stage. External TTS (also called parallel TTS in Zeng et al. (2025)) is proposed for selecting the correct reasoning process. Prominent external TTS algorithms include Best-of-N sampling, Beam Search, and Diverse Verifier Tree Search, using the reward model as the verifier to select high-quality reasoning trajectories. Researchers (Lightman et al., 2023) have shown that the Process Reward Model(PRM) is more effective in performance boosting compared with the Outcome Reward Model(ORM). However, Wang et al. (2023); Guan et al. (2025) point out that training a high-quality PRM remains costly, primarily due to the lack of accurate process-level annotations. Moreover, during the inference stage, introducing an additional LLM-based PRM introduces significant extra parameters and computational overhead, which severely limits the practical deployment of external TTS.

This paper focuses on external TTS and proposes a new Reflective Generative Form for high-quality reasoning trajectory selection. Specially,

the proposed new form shares the backbone of the policy model and process reward model, providing a more efficient scoring process with little parameter and computational overhead. Besides, a Self-supervised Process Reward Mode(SPRM) is introduced for self-supervised training to eliminate the reliance on process-level supervision. Based on the Reflective Generative Form, the proposed RGM contains high, medium, and low reasoning effort modes with the controllable thinking length. Experiment results show that RGM achieves comparable performance to OpenAI o3-mini's series with only 32B parameters.

In summary, the main contributions of this paper are as follows:

- We provide **a new Reflective Generative Form** for high-quality reasoning trajectory selection, which enables a single network to achieve both reasoning trajectory prediction and selection (with **Zero** process-level annotation).
- We provide both qualitative and quantitative analysis for **the aha moment, scaling law, and robustness of the proposed new form**. These exhaustive discussions will effectively benefit the community for future research.
- RGM achieves **comparable performance as the OpenAI o3-mini's series with only 32B parameters**, and outperforms a series of open-source and closed-source models.

2 Related Works

2.1 Test-Time Scaling

Test-Time Scaling (TTS) is a technique that leverages additional computational resources at inference time to tackle challenging problems. TTS can be divided into two categories: internal TTS and external TTS. Internal TTS introduces the long Chain-of-Thought (CoT) to generate answers based on the detailed reasoning process. OpenAI o1(Jaech et al., 2024) and DeepSeek R1(Guo et al., 2025) introduce a thinking process to plan the solution and guide the final answer. Jin et al. (2024); Yeo et al. (2025) have shown that long CoT can help models correct mistakes by themselves and decompose complex problems more effectively. However, Chen et al. (2024b,a) have highlighted the risk of overthinking, where excessively long reasoning trajectories may lead to performance degradation. On the other hand, external TTS scales up inference through search-based strategies and auxiliary reward models. A common approach is the Best-of-N strategy (Lightman et al., 2023; Brown et al., 2024; Wang et al., 2023). Fine-grained step level searching methods have also been explored, such as Beam Search (Liu et al., 2025; Snell et al., 2024), Diverse Verifier Tree Search (Beeching et al.) and Monte Carlo Tree Search (MCTS) (Zhang et al., 2024; Guan et al., 2025; Luo et al., 2024). These methods search at the step level and utilize Process Reward Models (PRMs) to guide the reasoning trajectory step-by-step. Beyond search strategies, recent work emphasizes that the quality of the reward model is a crucial factor in external TTS (Guan et al., 2025).

2.2 Process Reward Model

Process Reward Models (PRMs) focus on evaluating LLMs at the step level. Lightman et al. (2023) unveil that this fine-grained guidance can lead to better TTS performance compared with the global-level Outcome Reward Model (ORM). However, accurately identifying logical errors in LLM outputs remains challenging, and PRMs require high-quality task-specific annotated data for training. To this end, recent works Wang et al. (2023) leverage Monte Carlo estimation to automatically assign step-level scores using only the final answers as supervision. Zhang et al. (2024); Guan et al. (2025) iteratively synthesizes data by MCTS and fine-tuning both LLMs and PRMs, improving performance across both models. Tan et al. (2025) follow the LLM-as-a-judge method and introduce a new LLM to annotate the reward of each step. Nonetheless, Zhang et al. (2025) point out that labels generated by Monte Carlo estimation can be noisy, as incorrect reasoning processes may still yield correct final answers. They further propose a hybrid approach that combines both Monte Carlo estimation with the LLM-as-a-judge.

3 Problem Formulation

This paper aims to find a high-quality reasoning trajectory more efficiently at inference time based on TTS. We first summarize the general inference forms for standard LLMs (policy models) and existing TTS methods, and then formally define our proposed Reflective Generative Form.

1) Basic LLMs. The model directly generates an answer based on the input query Q. This basic in-

ference form can be formulated as:

$$\text{answer} = LLM_{\text{answer}}(\text{Q}). \quad (1)$$

TTS based methods can be categorized into two types: sequential scaling based internal TTS and parallel scaling based external TTS.

2) Internal TTS. The internal TTS first generates a reasoning trajectory by Long-CoT using LLM_{think}, and then predicts the final answer with this trajectory using LLM_{answer}, which can be expressed as:

$$\text{answer} = LLM_{\text{answer}}(LLM_{\text{think}}(\text{query})). \quad (2)$$

To be specific, recent methods (e.g. DeepSeek R1(Guo et al., 2025)) use the same policy model for both LLM_{think} and LLM_{answer}.

3) External TTS. Firstly, the Long-CoT generation is extended by generating multiple reasoning trajectories and answers in parallel. Then, a reward model (e.g. PRM) is used to score and select the best result (Lightman et al., 2023; Liu et al., 2025). This inference form can be described as:

$$\begin{aligned} \text{answer}_i &= [LLM_{\text{answer}}\big(LLM_{\text{think}}(\text{Q})\big)]_i. \\ \text{answer} &= \arg\max_{i\in[1,k]} LLM_{PRM}\big(\text{answer}_i\big), \end{aligned} \quad (3)$$

where $[*]_i$ denotes the i-th candidate among k parallel generations.

Though existing external TTS methods have been proven to obtain considerable performance enhancement, they still encounter several problems: (1) Extra Computation: PRM contains individual parameters from the policy model(LLM_{think} and LLM_{answer}), which introduces additional huge computation. (2) Expensive Annotation: It is difficult to obtain the large-scale reasoning trajectory annotations for PRM training.

Reflective Generative Form. To address the extra computation and expensive annotation issues, we propose a new Reflective Generative Form focusing on the efficient and label-free reasoning trajectory selection. The proposed Reflective Generative Form is shown in follows,

$$\begin{aligned} \text{think}_{i^*} &= \arg\max_{i\in[1,k]} [LLM^{share}_{\text{SPRM}}\big(LLM^{share}_{\text{think}}(\text{Q})\big)]_i, \\ \text{answer} &= LLM^{share}_{\text{answer}}\big(\text{think}_{i^*}\big). \end{aligned} \quad (4)$$

Firstly, we share the backbone of the policy model and PRM in a single network, which enables reasoning trajectory generation and scoring in a unified interface for parallel prediction. The score measures the quality of each reasoning trajectory, and the trajectory with higher score is selected as the high-quality candidate in TTS. This unified interface is proved to be effective for parameter reduction in our experiments. Secondly, we introduce a novel Self-supervised Process Reward Model (SPRM) to eliminate the reliance on process-level annotation, which can be optimized with only outcome-level annotation in a self-supervised manner. In particular, we only implement the SPRM for the LLM_{think} selection, which can further improve the inference efficiency during the real implementation.

4 Approach

4.1 Unified Interface in Reflective Generative Form

Our proposed Reflective Generative Form establishes a unified interface for the policy model and the PRM. For the policy model, we employ reasoning LLMs that contain the thinking process in response, delineated by the '<think>' and '</think>' tokens. For the PRM, we introduce a Self-supervised Process Reward Model (SPRM), which shares the same backbone as the policy model but incorporates an additional lightweight SPRM head. The SPRM head is implemented by a binary classifier consisting of two linear layers and a dropout layer. An overview of the joint framework is illustrated in Fig. 1(a).

Within this unified form, the policy model first generates multiple thinking processes as the reasoning trajectories. Subsequently, the SPRM evaluates each thinking process for reasoning trajectory selection. The evaluation procedure contains two steps: (1) Segmenting the reasoning trajectory into discrete steps and (2) Predicting a trajectory score based on evaluation in each step.

1. Step Segmentation. We segment each reasoning trajectory using tokens that are already supported by the policy model's tokenizer, eliminating the need to introduce additional step-specific tokens or fine-tune the LLM for step-format outputs. Specifically, we treat tokens containing '.\n\n' as step-tokens and split the trajectory accordingly. Additionally, we retain only the first token in any sequence of consecutive step-tokens and ignore the step-token appearing at the beginning of the trajectory, as it does not contain valuable information.

2. Trajectory Score Prediction. After using step-

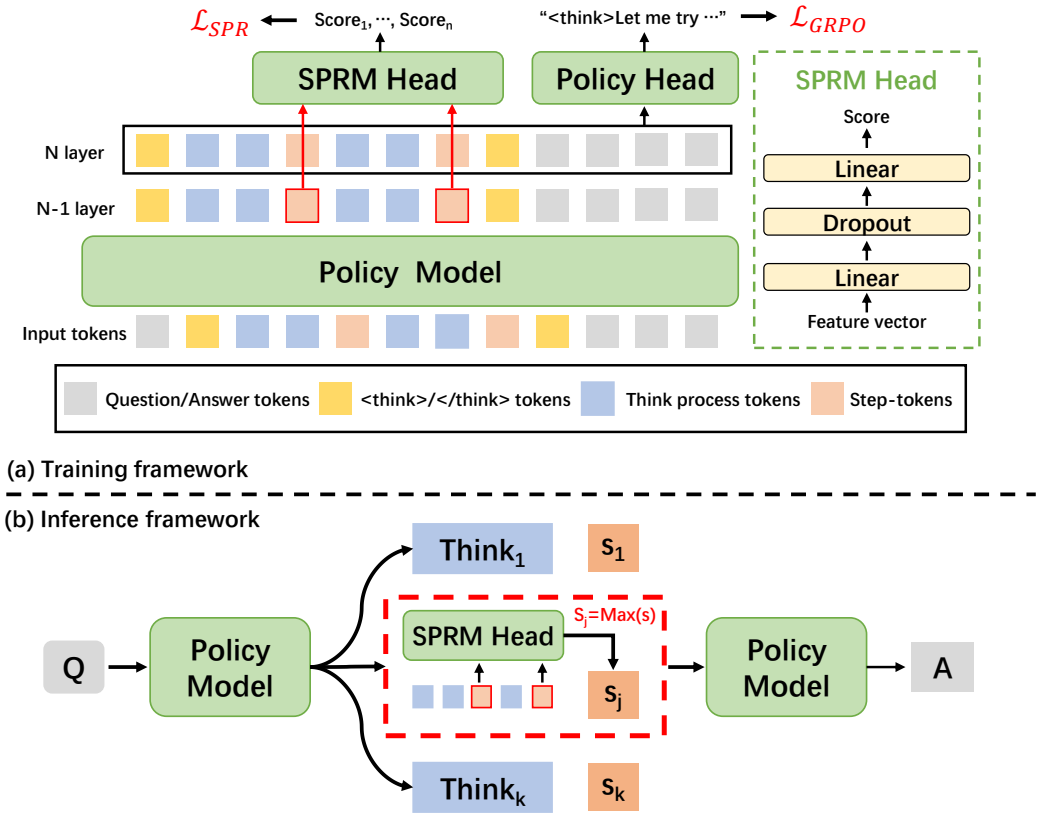

Figure 1: The training and inference framework of Reflective Generative Models.

tokens to mark the end of individual reasoning steps, we evaluate each step based on the representation of the corresponding step-token. Since the representation in the last layer mainly captures the logits prediction for a single token, we use the hidden representations from the second-to-last layer of the policy model to provide richer contextual information of the entire step. These representations are then fed into the SPRM head to predict process scores for each step. The final score for the entire reasoning trajectory is computed as the geometric mean of the individual process scores:

$$\begin{aligned} \mathrm{S}_{\text{final}} &= \left(\prod_{n=1}^{N} \mathrm{Score}_n \right)^{\frac{1}{N}} \\ &= \left(\prod_{n=1}^{N} SPRM(f_{token_n}) \right)^{\frac{1}{N}}, \end{aligned} \tag{5}$$

where N denotes the total number of steps, and f_{token_n} is the representation of the n-th step-token obtained from the policy model. Score_n is the SPRM's process score for n-th step.

Through this unified interface, a single network can generate reasoning trajectories and score them in parallel, enabling joint training in an end-to-end manner. This design facilitates a straightforward and efficient training pipeline for on-policy PRM learning, where both the policy model and the SPRM continuously refine their parameters from shared experiences, thereby improving the overall quality of the generated trajectories.

4.2 Optimization of Reflective Generative Form

During optimization, we train the policy model and the SPRM head simultaneously. For the policy model, we adopt Group Relative Policy Optimization (GRPO) following Shao et al. (2024). To optimize the SPRM head, we propose a Self-supervised Process Reward Loss (SPR Loss), which enables learning process discrimination ability only from outcome reward (e.g. final answer correctness). The SPR Loss is formulated as follows,

$$\mathcal{L}_{\text{SPR}} = \frac{1}{N} \sum_{n=1}^{N} w_n * BCELoss(\mathrm{Score}_n, y_n),$$
$$\text{where } w_n = \begin{cases} 1, & \text{if } y_n = 1 \ \& \ \mathrm{Score}_n > 0.5 \\ 1, & \text{if } y_n = 0 \ \& \ \mathrm{Score}_n < 0.5 \\ 0, & \text{others} \end{cases}, \tag{6}$$

where n denotes the step-tokens, w_n is a token-level weight, Score_n is SPRM's process score on step n, and y_n denotes whether the final answer from the policy model is correct. In Eq.6, the process score is optimized based on the correctness of the final answer. However, since a correct final answer may include incorrect intermediate steps and vice versa (Lightman et al., 2023), we introduce the self-supervised dynamic weight w_i to mitigate supervision noise. Specifically, we use the SPRM head's own prediction on each step as the pseudo label and set $w_n = 1$ only if the pseudo label is consistent with the final answer's correctness. This dynamic filtering allows the model to avoid noisy samples and focus on the most representative steps of correct and incorrect solutions. Thus, by enlarging the score gap between correct and incorrect steps, SPRM can progressively learn the process evaluation ability with only final annotations.

4.3 Inference with Reflective Generative Form

In the inference stage, our Reflective Generative Form is naturally suitable for TTS where the SPRM can provide guidance for selecting the high-quality reasoning trajectory from the policy model. The total inference process divides into three steps(shown in Fig.1(b)): (1) For the given question, the policy model first samples k thinking processes as the candidate reasoning trajectories: $think_1, think_2, \ldots, think_k$. (2) The SPRM evaluates the steps in each process and obtains the final score by the geometric mean of corresponding process scores: $S_1, S_2, \ldots, S_k$. (3) The reasoning trajectory with the highest final score is chosen and guides the policy model to answer the question (Eq.7).

$$\begin{aligned} i^* &= \arg\max_{i \in [1,k]} (S_1, \ldots, S_k), \\ \text{answer} &= LLM_{\text{answer}}(think_{i^*}). \end{aligned} \tag{7}$$

5 Experiment

5.1 Baseline&Dataset

We conduct experiments on the models with three different sizes: RGM-1.5B, 7B, and 32B, which are initialized from DeepSeek-R1-Distill-Qwen-1.5B/7B (Guo et al., 2025), and QWQ-32B (Team, 2025) with continual reinforcement training. After adding the SPRM head, only 5M/26M/53M extra parameters are introduced for RGM-1.5B/7B/32B. Our training dataset is constructed from multiple publicly available math-related sources, including NuminaMath (Li et al., 2024), OpenR1-Math-220k, DeepScaleR (Luo et al., 2025), LIMR (Li et al., 2025), and OREAL-RL (Lyu et al., 2025). We apply a multi-agent data cleaning framework to ensure data quality, resulting in a final dataset of 40k high-quality examples. The models are trained on 64 H200 GPUs with batch size of 128 and response length of 32k. In the inference stage, we use the sampling temperature of 0.6 and output length of 32k. We set 3 reasoning effort modes with $k = 2, 8, 32$ in Eq.7, named RGM-low, -medium, and -high.

We evaluate our models on 2 mathematical benchmarks: AIME2024 and AIME2025 (AIME, 2025). To verify the robustness of our RGM, we introduce 2 extra out-of-distribution benchmarks: LivecodeBench(240801-250201) (Jain et al., 2024) (for coding capability evaluation) and C-Eval (Huang et al., 2023) (for Chinese reasoning capability evaluation). We adopt Pass@1 as the evaluation metric. For each problem, the model generates only one final answer, and the Pass@1 score is computed as the proportion of correctly solved problems. To improve the stability of the results, we repeat the evaluation 64 times and report the average accuracy as the final score.

5.2 Main Results

Table 1 summarizes the performance of our RGM across fo4ur representative benchmarks. We denote baseline as the only policy model without using the Reflective Generative Form. Across different model scales, our proposed Reflective Generative Form consistently enhances the baseline, particularly on mathematical reasoning benchmarks. Specifically, compared with the baseline, RGM achieves performance gains of 18.6/15.5/5.3 points on AIME24 and 10.5/7.4/3.1 points on AIME25 for the 1.5B/7B/32B sizes, respectively. Our Reflective Generative Form is also robust on other out-of-domain tasks, yielding gains of 5.7/5.0/0.8 points on LiveCodeBench and 2.3/6.5/0.3 points on C-Eval.

We further compare RGM against both advanced open-source and closed-source models. Among the open-source models, we consider DeepScaleR-1.5B-Preview (Luo et al., 2025), DeepSeek-R1-Distill-Qwen (1.5B/7B/32B), DeepSeek-R1-Distill-LLaMA-8B (Guo et al., 2025), QwQ-32B (Team, 2025), GLM-Z1-32B-0414 (GLM et al., 2024), s1-32B (Muennighoff et al., 2025), and DeepSeek-R1-

Model	Mathematical		Out-of-Distribution	
	AIME24	AIME25	LiveCodeBench	C-Eval
Small-size Open-Source Models				
DeepScaleR-1.5B-Preview	43.1	30.0	-	-
R1-Distill-Qwen-1.5B	28.9	22.8	16.9	27.1
R1-Distill-Qwen-7B	55.5	-	37.6	-
R1-Distill-Llama-8B	50.4	-	39.6	-
Baseline-1.5B	39.3	29.9	22.4	41.8
RGM-1.5B-low	44.0	32.6	24.2	43.6
RGM-1.5B-medium	53.1	35.7	26.6	43.9
RGM-1.5B-high	57.9	40.4	28.1	44.1
Baseline-7B	54.7	41.2	39.4	51.3
RGM-7B-low	60.7	45.4	41.7	55.1
RGM-7B-medium	66.3	48.3	44.1	57.5
RGM-7B-high	**70.2**	**48.6**	**44.4**	**57.8**
Large-size Open-Source Models				
s1-32B	56.7	50.0	-	-
QwQ-32B	79.5	69.5	63.4	88.4
R1-Distill-Qwen-32B	72.6	49.6	57.2	82.2
GLM-Z1-32B-0414	80.8	63.6	59.1	-
DeepSeek-R1-671B	79.8	70.0	65.9	**91.8**
Closed-Source Models				
Claude-3.5-Sonnet1022	16.0	7.4	37.2	76.7
GPT-4o-0513	9.3	11.6	32.9	-
OpenAI o1-mini	63.6	50.7	53.8	68.9
OpenAI o1-1217	79.2	-	63.4	-
OpenAI o3-mini*	79.6	**74.8**	**67.4**	75.9
Baseline-32B	79.9	70.5	63.4	89.4
RGM-32B-low	82.0	72.0	63.8	89.5
RGM-32B-medium	84.2	73.4	64.0	89.6
RGM-32B-high	**85.2**	73.6	64.2	89.7

Table 1: Comparison of our RGM and other comparable models. * denotes medium level of OpenAI o3-mini. The best and second-best results are shown in **bold** and underlined.

671B (Guo et al., 2025). Among the closed-source models, we include Claude-3.5-Sonnet-1022, GPT-4o-0522, OpenAI o1-mini, OpenAI o1-1217, and OpenAI o3-mini-medium.

At the small scale, RGM-1.5B and RGM-7B consistently outperform the listed open-source models with comparable or larger parameter sizes. Especially, RGM-1.5B-low surpasses DeepScaleR-1.5B-Preview and R1-Distill-Qwen-1.5B on all datasets. And RGM-1.5B-high further outperforms R1-Distill-Qwen-7B and R1-Distill-Llama-8B on AIME24 (57.9% vs 55.5%), demonstrating strong efficiency and capability of our lightweight SPRM. For RGM-7B, its low reasoning effort mode has outperformed R1-Distill-Qwen-7B and R1-Distill-Llama-8B on AIME24 and LiveCodeBench. And RGM-7B-high further gains the improvement of 14.7 points on AIME24 (70.2% vs 55.5%) and 4.8 points on LiveCodeBench (44.4% vs 39.6%).

At the larger scale, RGM-32B-high achieves superior results on mathematical reasoning tasks, outperforming all listed open-source models of comparable or even larger size by +4.4% on AIME24 (85.2% vs 80.8%) and +3.6% on AIME25 (73.6% vs 70.0%). For other out-of-distribution tasks, RGM-32B-high surpasses other 32B-sized models

Model	Reward Model	Extra Params	AIME24	AIME25	LiveCodeBench
RGM-1.5B-high	Qwen2.5-Math-RM	72B	55.8	35.3	26.8
	Qwen2.5-Math-PRM	72B	56.7	40.0	26.8
	SPRM	**5M**	**57.9**	**40.4**	**28.1**
RGM-7B-high	Qwen2.5-Math-RM	72B	63.5	46.7	42.3
	Qwen2.5-Math-PRM	72B	68.8	48.3	42.8
	SPRM	**26M**	**70.2**	**48.6**	**44.4**

Table 2: Comparison of SPRM and other PRM models.

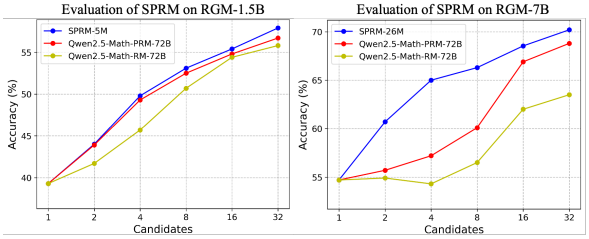

Figure 2: Evaluation of varying numbers of candidate reasoning trajectories on AIME24.

Model	Loss	AIME24	LiveCodeBench
RGM-1.5B-high	BCELoss	56.7	27.9
	SPRLoss	**57.9**	**28.1**
RGM-7B-high	BCELoss	69.1	43.9
	SPRLoss	**70.2**	**44.4**

Table 3: Evaluation on SPRLoss.

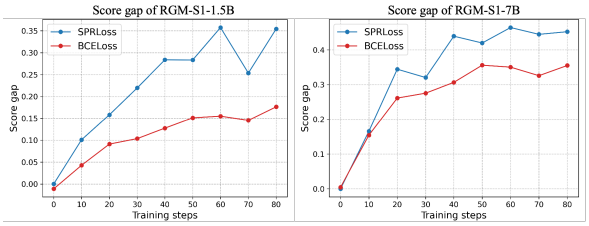

Figure 3: The prediction score gap between correct and incorrect solutions. The blue curve shows the SPRLoss. The red curve shows the BCELoss.

by +0.8% on LiveCodeBench (64.2% vs 63.4%) and +1.3% on C-Eval (89.7% vs 88.4%). Compared to closed-source models, medium and high levels of RGM-32B outperform Claude-3.5-Sonnet-1022 and GPT-4o-0522, and achieving comparable performance with medium level of OpenAI o3-mini (85.2% vs 79.6% on AIME24, 73.6% vs 74.8% on AIME25, 64.2% vs 67.4% on LiveCodeBench, 89.7% vs 75.9% on C-Eval,), highlighting its strong competitiveness in mathematical tasks and robustness on general tasks.

Besides, RGM with a more advanced backbone (Qwen3(Yang et al., 2025)) can be found in Appendix.A.2.

5.3 Ablation Study

Effectiveness of SPRM. As shown in Table.2, our SPRM, with only around 26M extra parameters, achieves higher performance compared with 72B ORM and PRM. This demonstrates that sharing parameters between the reward model and policy model is able to generate high-quality guidance without the requirement of additional large-scale reward models. To further analyze the efficacy of different reward models, we compare their performance under different numbers of candidate reasoning trajectories k in Fig.2. Across different k and model sizes, SPRM consistently outperforms other methods, indicating its strong ability to distinguish between high and low quality reasoning trajectories. The detailed predictions of SPRM are listed in Appendix.A.3

Moreover, we evaluate the generalization ability of SPRM on tasks from the other domain (LiveCodeBench). Without any task-specific fine-tuning or in-domain data, our SPRM still achieves superior performance compared to separate reward models. This demonstrates SPRM's strong zero-shot generalization capability, suggesting that it can capture domain-agnostic patterns to evaluate the reasoning trajectories.

Effectiveness of self-supervised optimization. We evaluate the effectiveness of SPRLoss in Table.3. Compared with using the final answer correctness as process-level supervision for PRM training, our proposed self-supervised optimization method achieves larger performance gains on both 1.5B and 7B models. Furthermore, Fig.3 shows the prediction score gap between correct and incorrect solutions. Compared to the BCELoss, SPRLoss demonstrates stronger discriminative capability with a larger score gap. This indicates that treating final answer correctness as process-level labels introduces substantial label noise, which harms the

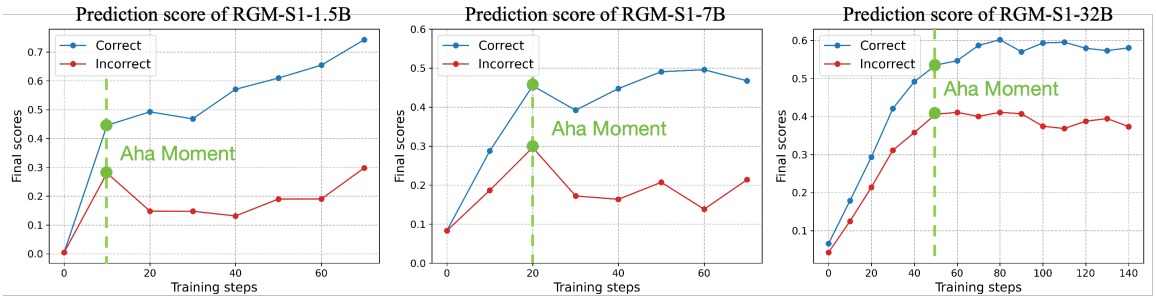

Figure 4: The training process of SPRM. The blue and red curves denote the final score on correct and incorrect reasoning trajectories. The green dashed line indicates the "aha moment".

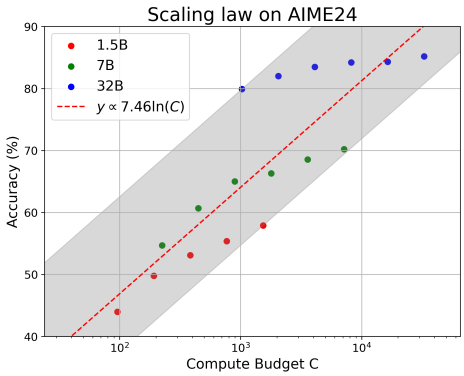

Figure 5: The scaling law of Reflective Generative Models.

optimization. In contrast, SPRLoss leverages self-supervised signals to reduce the impact of noisy supervision, leading to stable and accurate training.

5.4 Aha Moment of RGMs

In Fig. 4, we present the final evaluation scores from RGM for both correct and incorrect reasoning trajectories throughout the training process. During the initial phase of training, the optimization trajectories for all samples follow a similar trend, indicating that the model has not yet learned to distinguish between correct and incorrect reasoning trajectories. However, after a certain number of training steps (e.g., around 10/20/50 steps, 1280/2560/6400 samples for RGM-1.5B/7B/32B, respectively), we observe a distinct "aha moment" point where the optimization trends of different reasoning trajectories begin to diverge. This suggests that the model is starting to judge the correctness based on the reasoning contents. With this aha moment, RGM can progressively refine its SPRM head through our proposed self-supervised SPRLoss, leading to a clear score gap between correct and incorrect reasoning trajectories, and enabling effective TTS. The typical "aha moment" case study can be found in Appendix.A.3

5.5 Scaling Law of RGMs

In Fig. 5, we present the scaling law for reflective generative models, which shows the relationship between the total reasoning computation in TTS and the final performance. Following Snell et al. (2024), we define the computation budget C as the product of the model's parameter (B) and the total number of reasoning tokens (k): $C = Param_{policy} \times Token_{infer}$. Notably, when the total reasoning length is scaled to more than 32 times the baseline (e.g., Best-of-64), the performance improves slowly. Therefore, we mainly focus on TTS results up to Best-of-32 for each model scale. We observe that the final performance shows a positive correlation with the logarithm of the computation budget (the specific scaling factor depends on the baseline model architecture). This indicates that the final performance of our model can be enhanced by exponentially scaling on parameter size or the reasoning length.

6 Conclusion

In this work, we propose a novel Reflective Generative Form, which enables a single LLM to both generate and select high-quality reasoning trajectories for Test-Time Scaling (TTS). Based on this form, we present the reflective generative model (RGM). Specifically, we design a unified interface that integrates the policy model and process reward model (PRM) within a single network, resulting in low parameter overhead and efficient TTS inference. A self-supervised process reward model (SPRM) is proposed to learn process-level evaluation with only final answer annotations. With 32B parameters, RGM achieves comparable performance to the OpenAI o3-mini series across mathematics, coding, and Chinese reasoning benchmarks.

7 Limitations

Despite the promising performance, our Reflective Generative Model has several limitations. First, the best-of-N strategy requires the policy model to generate N complete reasoning trajectories before selecting the final output, which cannot be used for streaming output and may introduce additional latency. This is a common problem for Test-Time Scaling based methods. Second, the model's improvements on non-mathematical tasks remain limited. For the models on 32B level, the improvement on LiveCodeBench and C-Eval are only 0.8 and 0.3 points compared with the baseline. This is due to our training set only containing mathematical data. Building a large-scale training set including more domains could help address this issue. Third, we only use Qwen and QWQ-32B architectures as the backbone. The detailed step of aha moment and the specific growth rate of scaling law may be different for other architectures. Moreover, the effectiveness of our method on large-scale models with over 32B parameters remains to be explored. Finally, our model with QwQ-32B backbone still underperforms OpenAI o3-mini-medium on some benchmarks, such as AIME25 and LiveCodeBench. Using a stronger backbone could help mitigate this gap.

References

AIME. 2025. AIME problems and solutions.

Edward Beeching, Lewis Tunstall, and Sasha Rush. Scaling test-time compute with open models.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.

Qiguang Chen, Libo Qin, Jiaqi Wang, Jingxuan Zhou, and Wanxiang Che. 2024a. Unlocking the capabilities of thought: A reasoning boundary framework to quantify and optimize chain-of-thought. *Advances in Neural Information Processing Systems*, 37:54872–54904.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, and 1 others. 2024b. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*.

François Chollet. 2024. Openai o3 breakthrough high score on arc-agi-pub. Accessed: 2024-12-20.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, and 37 others. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *Preprint*, arXiv:2406.12793.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. 2025. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *arXiv preprint arXiv:2501.04519*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, and 1 others. 2023. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Advances in Neural Information Processing Systems*, 36:62991–63010.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. 2024. The impact of reasoning step length on large language models. *arXiv preprint arXiv:2401.04925*.

Adaline Labs. 2025. Inside reasoning models openai o3 and deepseek r1. Accessed: 2025-02-18.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, and 1 others. 2024. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13:9.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. 2025. Limr: Less is more for rl scaling. *arXiv preprint arXiv:2502.11886*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Runze Liu, Junqi Gao, Jian Zhao, Kaiyan Zhang, Xiu Li, Biqing Qi, Wanli Ouyang, and Bowen Zhou. 2025. Can 1b llm surpass 405b llm? rethinking compute-optimal test-time scaling. *arXiv preprint arXiv:2502.06703*.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, and 1 others. 2024. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2. Notion Blog.

Chengqi Lyu, Songyang Gao, Yuzhe Gu, Wenwei Zhang, Jianfei Gao, Kuikun Liu, Ziyi Wang, Shuaibin Li, Qian Zhao, Haian Huang, and 1 others. 2025. Exploring the limit of outcome reward for learning mathematical reasoning. *arXiv preprint arXiv:2502.06781*.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*.

OpenAI. 2024. Learning to reason with llms. Accessed: 2025-04-13.

OpenAI. 2025. Openai o3-mini evaluation. Accessed: 2025-01-31.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Xiaoyu Tan, Tianchu Yao, Chao Qu, Bin Li, Minghao Yang, Dakuan Lu, Haozhe Wang, Xihe Qiu, Wei Chu, Yinghui Xu, and 1 others. 2025. Aurora: Automated training framework of universal process reward models via ensemble prompting and reverse verification. *arXiv preprint arXiv:2502.11520*.

Qwen Team. 2025. Qwq-32b: Embracing the power of reinforcement learning.

Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2023. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. 2025. Demystifying long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2502.03373*.

Maxwell Zeff. 2024. Openai's o3 suggests ai models are scaling in new ways — but so are the costs. Accessed: 2024-12-23.

Zhiyuan Zeng, Qinyuan Cheng, Zhangyue Yin, Yunhua Zhou, and Xipeng Qiu. 2025. Revisiting the test-time scaling of o1-like models: Do they truly possess test-time scaling capabilities? *arXiv preprint arXiv:2502.12215*.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024. Rest-mcts*: Llm self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*, 37:64735–64772.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*.

A Appendix

A.1 Extend on MCTS

Since SPRM produces process scores for each step, our reflective generative models can be naturally used in step-level search-based TTS methods such as Monte Carlo Tree Search (MCTS). In our setup, instead of performing full simulations to the end of a reasoning trajectory, we directly use SPRM to estimate the value of each node during the search. This enables a more efficient evaluation at each expansion step. In the expanding stage, we expand 4 children for the selected node and generate 1024 tokens in each child node. To balance the computation cost, we set the maximum number of tokens in the MCTS process from 0 (without MCTS) to 160k for each question.

Table.4 presents the performance of MCTS on the AIME24 dataset. By increasing the maximum

number of searching tokens, the performance improved from 39.3 to 52.8, demonstrating the effectiveness of SPRM in providing high-quality step-level guidance in the reasoning stage. However, the performance of MCTS is still lower than the results under the Best-of-N strategy reported in Table 1. This is primarily due to the computational overhead inherent in tree-based search methods, which leads to incomplete searching in our setting. Nonetheless, the observed gains over the baseline validate the potential of our reflective generative models for integration with more advanced search-based inference methods.

Tokens(k)	0	40	80	120	160
Accuracy(%)	39.3	48.8	50.0	51.7	52.8

Table 4: Performance of RGM-1.5B with MCTS on AIME24.

A.2 Generalization on advanced architecture.

To further assess the performance upper bound of our Reflective Generative Form, we integrate it with a more capable open-source model, Qwen3-32B (Yang et al., 2025), denoted as RGM-Qwen3-32B. Experimental results are displayed in Table.5. Across all four benchmarks, RGM-Qwen3-32B consistently outperforms the original Qwen3-32B model. On mathematical reasoning tasks, it achieves a stable improvement of +3.2% on AIME24 and +4.4% on AIME25. Similar gains are observed on other general benchmarks, with +1.5% on LivecodeBench and +2.4% on C-Eval. These results demonstrate that RGM scales effectively with the capacity of the backbone model, highlighting its generalization potential on more advanced models.

Benchmark	Qwen3-32B	RGM-Qwen3-32B		
		low	medium	high
AIME24	81.4	81.7	83.8	**84.6**
AIME25	72.9	74.8	77.1	**77.3**
LivecodeBench	65.7	66.6	67.0	**67.2**
C-Eval	87.3	89.4	89.5	**89.7**

Table 5: Evaluation with Qwen3 backbone. The v5 version of LivecodeBench is used for comparison.

A.3 Case study

Case study of SPRM Fig.6 shows the visualization of step-wise evaluation scores from SPRM. It can be observed that SPRM effectively identifies low-quality processes generated by the policy model, including logical error (e.g. the misunderstanding of b in step 58 of example 1) and calculation error (e.g., the incorrect computation $9 \times 21 + 7 = 193$ in step 32 of example 2). SPRM assigns low scores to these low-quality steps. Since SPRM only outputs process scores, we additionally provide post-descriptions within the dashed boxes for better clarity.

Case study of aha moment. We show an example in Fig.7. We fix the reasoning trajectory and use RGM before and after the aha moment for scoring. In this case, the model mistakenly confuses ε_k and ε_{k+1}, resulting in an incorrect solution. Our model fails to recognize the error before the aha moment, while the model after the aha moment can correctly discriminate it.

A.4 Artifacts Statement

All models and datasets utilized in this study are publicly available and distributed under permissible licenses.

Example 1

Question: Among the 900 residents of Aimeville, there are 195 who own a diamond ring, 367 who own a set of golf clubs, and 562 who own a garden spade. In addition, each of the 900 residents owns a bag of candy hearts. There are 437 residents who own exactly two of these things, and 234 residents who own exactly three of these things. Find the number of residents of Aimeville who own all four of these things.

Reasoning Trajectories:

…

Step 5:
However, in this case, we have four sets: diamond rings (D), golf clubs (G), garden spade (S), and candy hearts (C).

Score: 1.00. Define variables.

…

Step 58:
The number of residents who own exactly two things is b = 437. This includes those who own exactly two things among D, G, S, and those who own exactly one thing in D, G, S and exactly one thing in C.

Score: 0.44. Post-description: Encounter logical error. As everyone owns C, b should not include those who own exactly two things among D, G, S.

…

Final score: 0.00

Example2

Question: Find the sum of all integer bases $b > 9$ for which 17_b is a divisor of 97_b.

Reasoning Trajectories:

…

Step 30:
Therefore, possible bases are 21 and 49.

Score: 1.00. Post-description: The model successfully solved the question and the correct answer is 21+49=70.

…

Step 32:
Let me check $b = 21$:
Compute (17_{21}) : $(1 * 21 + 7 = 28)$
Compute (97_{21}) : $(9 * 21 + 7 = 186 + 7 = 193)$
Check if 28 divides 193? Wait, no, because 28*7=196, which is more than 193. Wait, perhaps my earlier assumption was wrong.

Score: 0.04. Post-description: Encounter calculation error. $9 \times 21 + 7$ should be 196 rather than 193. This causes the model to discard 21.

…

Final score: 0.03

Figure 6: SPRM's predictions on reasoning trajectories. Only key steps are listed. Correct and error steps are marked in green and red.

Example of Aha Moment in SPRM

Question: Let the sequence of rationals $x_1, x_2, \ldots$ be defined such that $x_1 = \frac{25}{11}$ and $[x_{k+1} = \frac{1}{3}\left(x_k + \frac{1}{x_k} - 1\right)]$. x_{2025} can be expressed as $\frac{m}{n}$ for relatively prime positive integers m and n. Find the remainder when $m+n$ is divided by 1000.

Reasoning Trajectories:

...

Step:
Alternatively, perhaps make a substitution. Let me define $\mu_k = 1/\varepsilon_k$. Then,
$1/\varepsilon_{k+1} = -1/\varepsilon_k + (8/9)/{\varepsilon_k}^2$
Multiply both sides by ${\varepsilon_k}^2$:
$\varepsilon_k = -\varepsilon_k + (8/9)$
Thus,
$\varepsilon_k + \varepsilon_{k+1} = 8/9$
But no, not quite.

Score: 0.52 (before Aha Moment) -> 0.12 (after Aha Moment).
Post-description: Encounter calculation error, model confuses ε_k and ε_{k+1} during the simplification.

...

Figure 7: Comparison of SPRM's predictions before and after the aha moment. Only key steps are listed. The error steps are marked in red.