🔍 *TrajSelector*: Harnessing Latent Representations for Effective and Robust Best-of-N Selection in Large Reasoning Models

Anonymous ACL submission

Abstract

Test-time scaling (TTS) improves large language model reasoning by allocating additional inference compute. In external TTS, Best-of-N selection samples multiple reasoning trajectories and chooses one final answer, making the selector a key bottleneck. Existing process reward models (PRMs) are often adapted from step-quality evaluation and rely on human-designed criteria such as readability or coherence, which may not always align with answer correctness and can lead to unstable scaling as N grows. We introduce ***TrajSelector***, a Best-of-N framework that uses the sampler LLM's hidden states as self-reflective signals for process scoring. A lightweight verifier (0.6B–4B parameters) scores reasoning steps from these hidden states and is trained only with outcome supervision. Across six mathematical reasoning benchmarks, our 0.6B verifier outperforms majority voting by 4.61% and 7B PRMs by 4.31–12.21% in Best-of-32, while the 4B verifier reaches 64.05% accuracy and scales robustly as N increases from 1 to 64. Additional results on code generation and tabular reasoning further support the generality of latent-state process scoring.

1 Introduction

Large language models (LLMs) have made rapid progress in mathematical reasoning (Comanici et al., 2025; Shao et al., 2024; Wang et al., 2025a), partly driven by test-time scaling (TTS), which allocates additional compute during inference (Guo et al., 2025a; Muennighoff et al., 2025; Xia et al., 2025). TTS can deepen a single reasoning path through longer chain-of-thoughts (Wei et al., 2023; Yu et al., 2025a) or broaden inference by sampling multiple candidate solutions in parallel (Zheng et al., 2025; Wang et al., 2025b; Fu et al., 2025; Chollet, 2024).

This work focuses on external TTS, where the model generates N independent solutions and a selector chooses one final answer (Figure 1). External TTS is attractive because it preserves the base sampler and parallelizes naturally, but its gains are limited by the quality of the selector: once the candidate set contains both correct and incorrect solutions, the final performance depends on whether the selector can identify the reliable trajectory. Existing selectors often use PRMs to score candidate trajectories (Xia et al., 2024; Zou et al., 2025; Cui et al., 2025). However, many PRMs are adapted from step-quality evaluation and rely on human-designed criteria or step-level labels, making their scores sensitive to stylistic priors such as readability, formatting, or step length. This creates a subtle mismatch: a fluent reasoning step may still lead to a wrong answer, while a terse or unconventional step may be essential for the correct solution. These priors need not align with final answer correctness, which may contribute to unstable Best-of-N behavior as N changes (Figure 2).

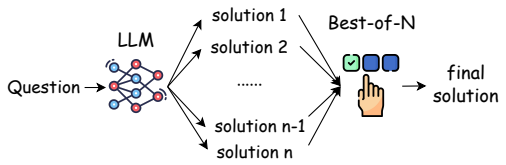


Figure 1: Best-of-N selection method illustration.

We propose TrajSelector, a Best-of-N framework that scores trajectories with the sampler LLM's own hidden states. Instead of imposing external step-quality criteria, TrajSelector trains a lightweight verifier to interpret step-final hidden states as self-reflective signals under outcome supervision. The verifier is therefore not asked to judge whether a step looks polished; it learns which latent patterns in the sampler correlate with correct final answers. This design avoids step-level human annotations, keeps the sampler frozen, and improves selection robustness as the candidate pool

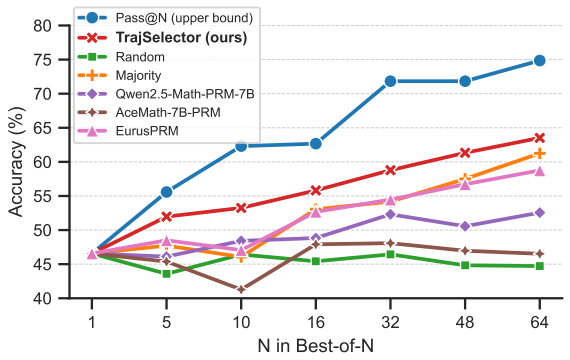


Figure 2: Best-of-N Scaling Curve. Existing PRMs can exhibit oscillatory behavior as N increases (e.g., performance drops from N=5 to N=10, then rises again), consistent with a mismatch between step-quality scoring criteria and final answer correctness. In contrast, TrajSelector achieves robust monotonic improvement by leveraging the sampler's intrinsic self-reflective signals.

grows. Conceptually, TrajSelector treats the sampler's latent representations as a compact trace of the generation process, allowing the selector to reuse information that is available during decoding but discarded by text-only PRMs.

We train *TrajSelector* from OpenR1-Math-220K (Hugging Face, 2025) and DeepMath-103K (He et al., 2025), freezing the sampler and updating only the verifier. As shown in Figure 2, TrajSelector improves robustly across $N \in [1, 64]$. In Best-of-32, the 0.6B verifier surpasses majority voting by 4.61% and 7B PRMs by 4.31–12.21%; scaling the verifier to 1.7B and 4B further improves accuracy to 61.33% and 64.05%. We further evaluate whether this latent-state selection mechanism transfers beyond the primary mathematical setting and whether the learned verifier can support downstream uses beyond Best-of-N inference.

Our main contributions are as follows:

- We show that mainstream PRMs can scale unstably in Best-of-N selection, a behavior consistent with possible mismatch between human-designed step-quality criteria and answer correctness.
- We present ***TrajSelector***, which uses sampler hidden states as self-reflective signals for lightweight process scoring and robust Best-of-N selection.
- We conduct a comprehensive evaluation on mathematical reasoning, further verify the adaptability of ***TrajSelector*** to code generation and tabular reasoning, and demonstrate its utility in two downstream tasks: offline data selection and RL reward modeling.

2 Related Work

2.1 Test-Time Scaling (TTS)

Test-Time Scaling improves reasoning by allocating additional compute at inference (Jaech et al., 2024). Internal TTS scales reasoning depth via extended chain-of-thought, as in OpenAI o1 (Jaech et al., 2024) and DeepSeek-R1 (Guo et al., 2025a). External TTS scales breadth by generating multiple trajectories for Best-of-N aggregation (Wang et al., 2023; Ma et al., 2025; Wang et al., 2025b; Zheng et al., 2025). While Wang et al. (2023) uses majority voting, Wang et al. (2025b) employs process scoring but requires RL-based sampler tuning, risking catastrophic forgetting. In contrast, our method keeps the sampler LLM entirely frozen.

2.2 Process Reward Model (PRM)

PRMs score reasoning trajectories for Best-of-N selection. Representative methods include Qwen2.5-Math-PRM-7B (Zhang et al., 2025b), PRM800K-tuned models (Zheng et al., 2024), and automated approaches like Math-Shepherd (Wang et al., 2024b) and ReasonEval-7B (Xia et al., 2024). ReasonFlux-PRM-7B (Zou et al., 2025) introduces trajectory-aware supervision, while EurusPRM (Cui et al., 2025) advances implicit process rewards. However, these PRMs were primarily designed to assess qualitative aspects such as readability, coherence, or formatting rather than Best-of-N selection. When such criteria diverge from final answer correctness, the resulting scores may become less reliable as the candidate pool changes. Our approach instead leverages the sampler's intrinsic self-reflective signals and trains the verifier directly from outcome supervision.

3 Problem Statement

Given a query x, an LLM $\mathcal{M}$ generates a final answer $\hat{r}$ with intermediate reasoning steps $\tau = (s_1, s_2, \ldots, s_T)$:

$$\tau \sim \mathcal{M}(\cdot|x), \quad \hat{r} \sim \mathcal{M}(\cdot|\tau, x) \tag{1}$$

We define correctness as $y = \mathbb{I}[r = g(\hat{r})]$, where $g(\cdot)$ extracts the answer from $\hat{r}$ and r is the ground

truth. In Best-of-N, the model generates N independent responses $\{\hat{r}_n\}_{n=1}^{N}$ with labels $\{y_n\}_{n=1}^{N}$, succeeding if any $y_n = 1$.

4 Method

Key motivations. TrajSelector targets three limitations of existing external TTS selectors: misaligned human-designed scoring criteria, underused sampler hidden states, and noisy step-level supervision. First, text-based PRMs often inherit human preferences about what reasoning should look like, which can diverge from what actually leads to a correct answer. Second, the sampler's hidden states contain generation-time information about confidence, uncertainty, and local reasoning dynamics, but this information is largely lost once the trajectory is decoded into text. Third, collecting reliable step-level labels is expensive, and automated step annotation can introduce additional noise. These observations lead to three design choices: keep the sampler frozen, reuse its step-final hidden states for scoring, and train the verifier from outcome labels rather than step annotations. A buffer-aware classification loss further improves robustness under noisy pseudo-labels.

4.1 Overview

Figure 3(a) summarizes ***TrajSelector***. Given a query, a frozen sampler $\mathcal{M}_\phi$ generates N candidate responses and exposes their hidden states. TrajSelector then follows a three-stage pipeline. First, candidate trajectories are sampled in parallel while their generation-time representations are retained. Second, each reasoning trajectory is segmented into steps, and the step-final hidden states are fed to a lightweight process score model f_θ. Third, the resulting step scores are averaged into a trajectory score, and the highest-scoring response is selected as the Best-of-N output. Because the verifier learns from the sampler's own hidden states under outcome supervision, scoring remains aligned with generation rather than external stylistic priors.

4.2 Process Score Model

The process score model (Figure 3(b)) consists of a tiny LLM initialized from Qwen3-0.6B-Base and a two-layer score head. Motivated by evidence that hidden states encode self-rewarding and self-reflective signals (Guo et al., 2025b; Fu et al., 2025), it scores sampler hidden states rather than generated text. This design differs from conventional PRMs in that the verifier does not need to re-parse the full textual trajectory from scratch; instead, it directly consumes the sampler's latent representation of each reasoning step. A linear projection maps the sampler hidden dimension to the verifier input space when needed, allowing the verifier to consume representations from larger samplers. The score head predicts three classes, *wrong*, *neutral*, and *right*; the step score is the probability of *right*, while *neutral* absorbs noisy supervision.

Step segmentation and aggregation. We extract the reasoning span between `<think>` and `</think>`, split it by double newlines, and use the final token hidden state of each step for scoring. This simple delimiter avoids requiring special step tokens or sampler fine-tuning, making the procedure compatible with standard reasoning traces. The resulting step scores are averaged to obtain a trajectory score, and the response with the highest score is selected as the Best-of-N output.

4.3 Training

High-quality step labels are scarce and can encode human priors about what "good" reasoning should look like. We therefore avoid manually designed step-level supervision and train with only the final outcome label, following the spirit of FreePRM (Sun et al., 2025). This choice makes the training signal directly tied to answer correctness, which is the quantity needed by Best-of-N selection. For a trajectory $\tau = (s_1, \ldots, s_T)$ and correctness label $y \in \{0, 1\}$, each step receives the pseudo-label in Equation 2.

$$\tilde{y}_t = y, \quad t = 1, 2, \ldots, T \tag{2}$$

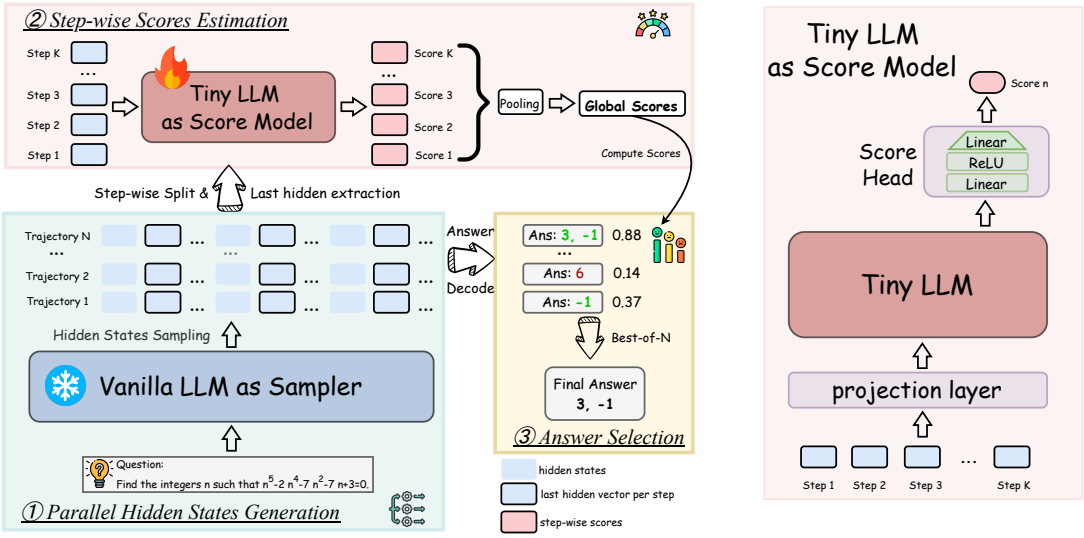
Because outcome-derived pseudo-labels are noisy, we introduce a buffer class (Sun et al., 2025). For each step s_t, the verifier predicts right (p_t^r), wrong (p_t^w), and buffer (p_t^b):

$$\begin{aligned} (p_t^r, p_t^w, p_t^b) &= f_\theta(s_t) \\ s.t.\ p_t^r + p_t^w + p_t^b &= 1 \end{aligned} \tag{3}$$

Given pseudo-label $\tilde{y}_t$, training encourages:

$$\begin{cases} p_t^r + p_t^b = 1, & \text{if } \tilde{y}_t = 1, \\ p_t^w + p_t^b = 1, & \text{if } \tilde{y}_t = 0. \end{cases} \tag{4}$$

The buffer class lets the model express uncertainty on ambiguous steps instead of overfitting noisy pseudo-labels. This is important because a correct final answer may contain redundant or imperfect intermediate steps, while an incorrect



(a) Overview of method **TrajSelector** (b) Process Score Model Architecture

Figure 3: **TrajSelector Framework Overview.** (a) The three-stage pipeline: parallel response generation with hidden state extraction, step-wise scoring via a lightweight process score model, and Best-of-N selection based on aggregated scores. (b) The process score model architecture, comprising a tiny LLM that processes hidden states and a score head that produces step-level quality estimates.

trajectory may still include locally valid reasoning. Forcing every step to inherit the final outcome as a hard binary label can therefore make the verifier sensitive to spurious correlations. The final loss is:

$$\mathcal{L}(\theta|\tau) = -\frac{1}{T}\sum_{t=1}^{T}[\tilde{y}_t \log(p_t^r + p_t^b) + (1-\tilde{y}_t)\log(p_t^w + p_t^b)] \tag{5}$$

The sampler $\mathcal{M}_\phi$ remains frozen throughout training.

5 Experiment

5.1 Experiment Settings

Training Dataset. We construct the training corpus from two public datasets: OpenR1-Math-220k (Hugging Face, 2025) and DeepMath-103K (He et al., 2025). Each example in the public datasets contains a question and a ground-truth answer. We employ Qwen3-8B (Team, 2025) to perform reasoning and generate the corresponding thinking trajectory and response for each question. Each response is automatically labeled by comparing its final answer with the ground truth via `Math-Verify` (Kydlíček). Given the substantial imbalance where correct samples outnumber incorrect ones, we retain all incorrect samples as negatives and downsample correct samples to a 1:1 ratio for positives. The resulting dataset is then shuffled to form the final training set which contains 133K examples. Statistical information can be found in Appendix B.

Network Architecture. We employ Qwen3-8B as the frozen sampler model and initialize the process score model using Qwen3-Base models of varying scales (0.6B, 1.7B, or 4B parameters). Hidden states mapping between the sampler and verifier is implemented through a linear layer, and the score head consists of two linear layers with a ReLU activation function (Agarap, 2019). Appendix E illustrates the network architecture pseudo-code.

Training Strategy. We employ DeepSpeed (Rajbhandari et al., 2020) and HuggingFace transformers (Wolf et al., 2020) to train the ***TrajSelector***. The hyper-parameters are shown in Appendix D.

Benchmarks. We conduct Best-of-N experiments on the following benchmarks: AMC23, AIME24, AIME25, HMMT25, BRUMO25 (Balunović et al., 2025) and BeyondAIME (Seed et al., 2025). We prompt the sampler model to generate N candidate responses in parallel for a given question, then employ different methods to select one response from the candidates as the final answer, and subsequently evaluate the correctness of this final answer to compute the accuracy metric. The user instruction used for generating candidate responses is shown in Appendix A.

Baselines. We compare our method with the following baseline methods: (1) **Random Selection**: Select a response randomly from the N candidates. (2) **Majority Voting** (Wang et al.,

Method	AMC-23 (/40)	AIME-24 (/30)	AIME-25 (/30)	BeyondAIME (/100)	HMMT-25 (/30)	BRUMO-25 (/30)	Avg (%)
			Best-of-32				
Pass@32 (Oracle)	**38**	**24**	**22**	**46**	**15**	**23**	**71.83**
Random Selection	34	17	12	17	8	16	46.44
Majority Voting	36	20	17	25	8	**18**	54.17
ReasonFlux-PRM-7B	35	19	16	21	8	16	50.86
Qwen2.5-Math-PRM-7B	35	**21**	16	23	8	16	52.31
Qwen2.5-Math-7B-PRM800K	36	19	15	20	7	15	49.44
ReasonEval-7B	35	20	15	20	7	**18**	51.25
Math-Shepherd	34	11	16	23	7	14	46.67
AceMath-7B-RM	35	19	13	21	6	16	48.08
EurusPRM	36	19	17	28	10	17	54.47
TrajSelector + 0.6B PRM (ours)	**38**	**21**	**18**	**31**	**11**	**18**	**58.78**
TrajSelector + 1.7B PRM (ours)	**38**	**22**	**19**	**33**	**12**	**19**	**61.33**
TrajSelector + 4B PRM (ours)	**38**	**23**	**20**	**36**	**13**	**20**	**64.05**
			Best-of-16				
Pass@16 (Oracle)	**38**	**24**	**22**	**41**	**15**	**21**	**62.67**
Random Selection	33	19	13	20	6	13	45.42
Majority Voting	36	**20**	**16**	25	9	16	53.06
ReasonFlux-PRM-7B	35	17	13	22	9	**19**	50.47
Qwen2.5-Math-PRM-7B	36	17	13	23	6	18	48.83
Qwen2.5-Math-7B-PRM800K	35	17	14	19	9	18	49.97
ReasonEval-7B	36	**20**	13	22	7	16	49.97
Math-Shepherd	34	18	12	22	5	16	46.17
AceMath-7B-RM	35	18	13	20	7	16	47.91
EurusPRM	36	19	13	26	**10**	18	52.67
TrajSelector + 0.6B PRM (ours)	**37**	**20**	**16**	**29**	**10**	18	**55.81**
TrajSelector + 1.7B PRM (ours)	**37**	**21**	**18**	**31**	**11**	**19**	**58.91**
TrajSelector + 4B PRM (ours)	**38**	**22**	**20**	**34**	**12**	**19**	**62.06**

Table 1: Experimental Results of Best-of-32 & Best-of-16

2023): Select the answer that appears most frequently as the final answer. (3) **Process Reward Model**: Select the response with the highest score computed by process reward model from the N candidates. We selected the following strong process reward models as baselines: Qwen2.5-Math-PRM-7B (Zhang et al., 2025b), Qwen2.5-Math-7B-PRM800K (Zheng et al., 2024), ReasonFlux-PRM-7B (Zou et al., 2025), ReasonEval-7B (Xia et al., 2024), Math-Shepherd (Wang et al., 2024b), AceMath-7B-RM (Liu et al., 2024) and EurusPRM (Cui et al., 2025; Sun et al., 2025). (4) **Pass@N**: A test is considered passed if at least one of the N candidate responses is correct. The results of this method represent the theoretical upper bound for Best-of-N selection.

5.2 Experiment Results

The main experimental results are presented in Table 1 and Table 2.

Effectiveness. TrajSelector consistently outperforms baselines across the six mathematical reasoning benchmarks and across Best-of-N settings. In Best-of-32, even the smallest 0.6B verifier achieves 58.78% average accuracy, outperforming Majority Voting by 4.61% and 7B PRM-based methods by 4.31% to 12.21%. These gains are especially meaningful because the verifier is substantially smaller than the 7B PRM baselines while using the same sampled candidate set. Scaling the verifier to 1.7B and 4B further improves performance to 61.33% and 64.05%, showing that TrajSelector benefits from additional verifier capacity while preserving a favorable accuracy-compute trade-off.

Robustness Analysis. Existing PRMs often exhibit oscillatory performance as N changes; for example, EurusPRM drops from 48.50% at $N = 5$ to 47.06% at $N = 10$, then rises to 52.67% at $N = 16$. Such behavior is undesirable for external TTS because larger candidate pools should make selection easier rather than less predictable. In contrast, TrajSelector improves monotonically with increasing N across all three verifier scales. This stability suggests that sampler hidden states provide a selection signal that is less sensitive to external stylistic preferences and more directly tied to answer correctness.

Computational Overhead. While our primary contribution lies in achieving robust and effective Best-of-N selection, we note that reusing hidden states also yields computational benefits as a side effect. Taking our 0.6B verifier as an ex-

Method	AMC-23 (/40)	AIME-24 (/30)	AIME-25 (/30)	BeyondAIME (/100)	HMMT-25 (/30)	BRUMO-25 (/30)	Avg (%)
			Best-of-1				
Pass@1	34	16	11	21	8	17	46.56
			Best-of-5				
Pass@5 (Oracle)	**37**	**21**	**14**	**31**	**11**	**17**	**55.58**
Random Selection	33	17	9	19	8	14	43.58
Majority Voting	36	15	**13**	23	7	**17**	47.72
ReasonFlux-PRM-7B	36	17	12	20	5	16	46.11
Qwen2.5-Math-PRM-7B	36	17	12	20	5	16	46.11
Qwen2.5-Math-7B-PRM800K	36	17	**13**	20	5	16	46.67
ReasonEval-7B	34	18	12	24	**9**	16	48.72
Math-Shepherd	36	16	11	20	5	15	44.44
AceMath-7B-RM	36	17	**13**	19	4	15	45.38
EurusPRM	36	18	**13**	21	7	16	48.50
TrajSelector + 0.6B PRM (ours)	**37**	**19**	**13**	**26**	**9**	**17**	**51.97**
TrajSelector + 1.7B PRM (ours)	**37**	**19**	**14**	**27**	**9**	**17**	**52.69**
TrajSelector + 4B PRM (ours)	**37**	**20**	**14**	**29**	**9**	**17**	**53.58**
			Best-of-10				
Pass@10 (Oracle)	**37**	**21**	**19**	**38**	**12**	**21**	**62.31**
Random Selection	35	16	13	21	4	18	46.42
Majority Voting	36	**18**	12	23	4	15	46.06
ReasonFlux-PRM-7B	35	**18**	12	23	5	**19**	48.42
Qwen2.5-Math-PRM-7B	35	**18**	13	23	5	18	48.42
Qwen2.5-Math-7B-PRM800K	**37**	17	13	22	6	18	49.08
ReasonEval-7B	36	16	**15**	19	6	15	47.06
Math-Shepherd	35	17	11	21	5	17	45.86
AceMath-7B-RM	34	15	12	16	4	13	41.28
EurusPRM	36	17	13	19	4	18	47.06
TrajSelector + 0.6B PRM (ours)	**37**	**18**	**15**	**27**	**9**	18	**53.25**
TrajSelector + 1.7B PRM (ours)	**37**	**19**	**16**	**29**	**11**	**19**	**56.36**
TrajSelector + 4B PRM (ours)	**37**	**20**	**18**	**31**	**11**	**20**	**58.92**

Table 2: Experimental Results of Best-of-1, Best-of-5 and Best-of-10

ample: (i) **Reduced GPU memory**. Compared with 7B PRMs, our method reduces PRM memory from 4,330M to 565.5M. (ii) **Faster inference**. The PRM inference time is reduced from 54.99s to 7.02s on a 30-question evaluation set. Even our largest 4B verifier remains more efficient than 7B PRMs while achieving superior performance (64.05% vs. 54.47% for the best 7B baseline EurusPRM in Best-of-32).

5.3 Comparison with Additional Selection Baselines

To further broaden the baseline coverage, we compare TrajSelector with two additional families of Best-of-N selection methods. The first family consists of **likelihood-based methods**, represented by DeepConf (Fu et al., 2025) and SWIFT (Guo et al., 2025c), which use token likelihood or confidence signals from the generation process to select candidate responses. The second family treats a **reasoning model as a process verifier**, represented by GenPRM (Zhao et al., 2025a) and R-PRM (She et al., 2025). These methods ask a reasoning model to evaluate intermediate reasoning steps and assign scores, providing strong but computationally expensive selection baselines. We obtain their results by reproducing the official implementations released by the original authors, using the default configurations provided in the official codebases under our evaluation setting.

As shown in Table 3, TrajSelector remains competitive against these additional selection baselines. The 0.6B verifier is comparable to the reasoning-model-as-PRM baselines while requiring a much smaller verifier, and the 4B verifier achieves the best average accuracy of 62.06%, outperforming the strongest additional baseline GenPRM by 5.37 percentage points. These results indicate that latent-state process scoring is competitive not only with conventional PRMs, but also with likelihood-based selectors and reasoning-model-based process verifiers.

5.4 Generalization Beyond Mathematical Reasoning

Although the main experiments focus on mathematical reasoning, we further evaluate *TrajSelector* on two additional strong reasoning domains: **code generation** and **tabular reasoning**. These tasks test different reasoning abilities from olympiad-

Method	AMC-23 (/40)	AIME-24 (/30)	AIME-25 (/30)	BeyondAIME (/100)	HMMT-25 (/30)	BRUMO-25 (/30)	Avg (%)
DeepConf	35	19	16	27	9	16	52.42
SWIFT	35	19	14	22	8	15	49.37
GenPRM	37	21	16	31	10	18	56.69
R-PRM	36	20	15	27	9	16	53.39
***TrajSelector* + 0.6B PRM (ours)**	**37**	20	**16**	29	**10**	**18**	55.81
***TrajSelector* + 4B PRM (ours)**	**38**	**22**	**20**	**34**	**12**	**19**	**62.06**

Table 3: Comparison with additional Best-of-16 selection baselines using Qwen3-8B as the sampler model. Results are reported as accuracy percentages.

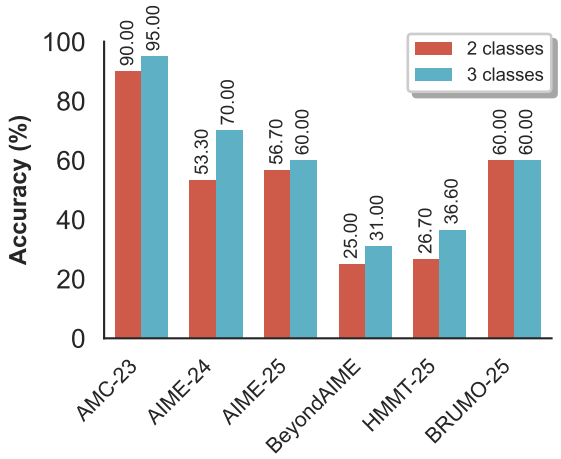


Figure 4: Ablation Study on Loss Function

style mathematics, including executable program synthesis and structured evidence aggregation over tabular inputs. They also stress different selection mechanisms: code generation cannot rely on simple majority voting because correct programs may be semantically equivalent but textually different, while tabular reasoning often requires selecting responses with faithful evidence aggregation rather than only matching a short final answer. The results show that *TrajSelector* remains effective beyond mathematics, suggesting that latent-state process scoring provides a general Best-of-N selection signal across reasoning-intensive domains. Detailed experimental settings and results are provided in Appendix J.

5.5 Ablations & Analysis

Ablation Study on Loss Function. To absorb noise in outcome-derived pseudo-labels, we replace standard binary classification with a three-class score head that includes a buffer class. We compare this design with a binary score head trained using standard BCELoss (Mao et al., 2023). Figure 4 shows that the buffer probability mitigates training noise and improves performance.

Ablation Study on Verifier Scale. We evaluate TrajSelector with three different verifier scales: 0.6B, 1.7B, and 4B parameters. As shown in Tables 1 and 2, performance consistently improves with larger verifiers across all Best-of-N settings. In Best-of-32, scaling from 0.6B to 4B improves accuracy from 58.78% to 64.05% (+5.27%), demonstrating that TrajSelector benefits from increased capacity while remaining significantly more efficient than 7B PRMs. Importantly, all three scales maintain robust monotonic scaling behavior as N increases, validating that our core design principle—leveraging intrinsic self-reflective signals—is effective across different model capacities.

Ablation Study on Sampler Model. Beyond Qwen3-8B, we also evaluate Qwen3-4B and Qwen3-14B as frozen samplers. Appendix H presents the detailed results, which remain consistent with our main conclusions.

Larger N in Best-of-N. The main experiments cover common parallel sampling budgets; we further evaluate larger N values in Appendix I. The results show that TrajSelector remains effective as the candidate pool grows.

More Model Families. To test generalization across LLM families, we replace the Qwen3 sampler with DeepSeek-R1-Distill-Qwen-7B and DeepSeek-R1-Distill-Llama-8B (Guo et al., 2025a). Appendix K shows that TrajSelector remains effective across these model families.

5.6 Downstream Application

Offline Data Selection. To demonstrate that the trained process score model can distinguish high-quality and low-quality reasoning trajectories, we apply TrajSelector to offline data selection. Specifically, TrajSelector scores candidate reasoning traces and selects high-quality samples for supervised fine-tuning. The selected data yields the best downstream SFT performance among all

compared selection strategies, achieving 86.4 on MATH500, 43.3 on AIME24, 43.3 on AIME25, and 53.5 on GPQA. These results indicate that TrajSelector can effectively re-rank reasoning trajectories for training data selection, showing that its process scores are useful not only for inference-time selection but also for constructing stronger training data. More details can be found in Appendix F.

RL reward modeling. We explore the application of TrajSelector as a lightweight verifier to provide reward signals for LLM reinforcement learning. Outcome-only rewards are sparse because they provide feedback only after a full solution is generated, making it difficult to identify which intermediate steps should be reinforced. We therefore treat the process scores generated by TrajSelector as step-level reward signals and integrate them into GRPO (Guo et al., 2025a) and DAPO (Yu et al., 2025b). The resulting hybrid reward improves both algorithms across all evaluated benchmarks; for example, DAPO+TrajSelector improves over DAPO from 79.0 to 82.3 on AIME24 and from 58.4 to 60.8 on BRUMO25. These results suggest that latent-state process scoring can serve as a practical reward-shaping signal in addition to its role as a Best-of-N selector. Detailed experimental setup and analysis are provided in Appendix G.

6 Discussion

Advantages over Other PRMs. Conventional PRMs usually receive only generated text, so the information flow from the sampler to the verifier is mediated by tokens and by the verifier's own learned priors. TrajSelector instead passes hidden states from the sampler to the verifier, preserving internal generation-time signals that are otherwise discarded after decoding. This design gives the verifier a more direct view of the sampler's reasoning process and reduces reliance on external human-designed step-quality criteria.

Why Does the Method Work? We attribute TrajSelector's effectiveness to three factors. First, checking whether a trajectory is likely to be correct is easier than solving the problem from scratch, so a smaller verifier can be sufficient. Second, the sampler's hidden states encode useful self-reflective and uncertainty-related signals (Wang et al., 2025b; Park et al., 2025; Guo et al., 2025c; Fu et al., 2025; Zhao et al., 2025b), allowing the verifier to reuse information already produced during generation. Third, outcome supervision avoids imposing stylistic step-level preferences; the buffer class then absorbs ambiguous pseudo-labels instead of forcing every step into a hard right/wrong decision.

Broader Implication. The results suggest that Best-of-N selection should not be viewed only as a post-hoc text ranking problem. During generation, the sampler already computes rich latent representations that reflect both the produced content and the model's internal state at each reasoning step. Text-only selectors discard this information and then attempt to reconstruct the quality of a trajectory from the decoded string. TrajSelector instead exposes this latent information to a lightweight verifier, turning the selector into a model-aware component of the inference pipeline. This perspective also explains why the same verifier can support downstream data selection and reward modeling: both applications require identifying trajectories whose reasoning process is likely to be useful, not merely trajectories whose final text looks polished.

Why not use BERT? Encoder-only models such as BERT (Devlin et al., 2019) are natural candidates for regression-style scoring, but LLM-generated reasoning trajectories often exceed BERT's 512-token context limit in our setting. We therefore adopt a tiny LLM as the verifier backbone, following recent progress in LLM-based scoring and embedding models (Zhang et al., 2025b,a; Wang et al., 2024a).

7 Conclusion

We introduced ***TrajSelector***, a framework for robust Best-of-N selection. Unlike PRMs that rely on human-designed step-quality criteria, TrajSelector uses the sampler's hidden states as self-reflective signals and trains a lightweight verifier with outcome supervision. Experiments show stable scaling behavior and strong accuracy: in Best-of-32, the 0.6B verifier outperforms majority voting by 4.61% and 7B PRMs by 4.31–12.21%, while 1.7B and 4B verifiers further improve performance to 61.33% and 64.05%. Additional results on code generation, tabular reasoning, offline data selection, and RL reward modeling suggest that latent-state process scoring is a broadly useful mechanism for reasoning-oriented selection and supervision.

Limitations

Our work primarily focuses on Best-of-N selection within external test-time scaling. Integrating latent-

state process scoring with search-based inference methods such as Monte Carlo tree search remains underexplored. Because TrajSelector relies on sampler hidden states, it requires white-box or at least representation-level access to the sampler model and is not directly applicable to purely black-box API models that expose only generated text. Although we add supplementary evaluations on code generation and tabular reasoning, the main training and evaluation protocol is still centered on domains with relatively clear automatic correctness signals; extending TrajSelector to open-ended question answering and subjective generation tasks requires further study. Finally, our use of TrajSelector for data selection and reinforcement learning is preliminary, and more refined training recipes remain future work.

Ethics Statement

The datasets and models used in this study are open-source, including OpenR1-Math-220K (Hugging Face, 2025), DeepMath-103K (He et al., 2025), and the Qwen series (Team, 2025). Our experiments focus on reasoning tasks with automatic evaluation signals, such as mathematical reasoning, code generation, and tabular reasoning, and do not involve human-subject data. As with other selection and reward modeling techniques, TrajSelector could amplify biases present in the sampler or training data if applied to open-ended user-facing domains; such applications should include domain-specific safety evaluation.

References

Abien Fred Agarap. 2019. Deep learning using rectified linear units (relu). *Preprint*, arXiv:1803.08375.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *Preprint*, arXiv:2108.07732.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. 2025. Matharena: Evaluating llms on uncontaminated math competitions. *Preprint*, arXiv:2505.23281.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021a. Evaluating large language models trained on code.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021b. Finqa: A dataset of numerical reasoning over financial data. *Proceedings of EMNLP 2021*.

François Chollet. 2024. Openai o3 breakthrough high score on arc-agi-pub. https://arcprize.org/blog/oai-o3-pub-breakthrough. Accessed: 2024-12-20.

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

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szpektor, Nan-Jiang Jiang, and 3290 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *Preprint*, arXiv:2507.06261.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, and 1 others. 2025. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. 2025. Deep think with confidence. *arXiv preprint arXiv:2508.15260*.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, Ashima Suvarna, Benjamin Feuer, Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, and 31 others. 2025. Openthoughts: Data recipes for reasoning models. *Preprint*, arXiv:2506.04178.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025a. Deepseek-r1: Incentivizing reasoning capability in

llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Jizhou Guo, Zhaomin Wu, Hanchen Yang, and Philip S. Yu. 2025b. Mining intrinsic rewards from llm hidden states for efficient best-of-n sampling. *Preprint*, arXiv:2505.12225.

Jizhou Guo, Zhaomin Wu, Hanchen Yang, and Philip S. Yu. 2025c. Mining intrinsic rewards from llm hidden states for efficient best-of-n sampling. *Preprint*, arXiv:2505.12225.

Nathan Habib, Clémentine Fourrier, Hynek Kydlíček, Thomas Wolf, and Lewis Tunstall. 2023. Lighteval: A lightweight framework for llm evaluation.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. 2021a. Measuring coding challenge competence with apps. *NeurIPS*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. *Preprint*, arXiv:2103.03874.

Hugging Face. 2025. Open r1: A fully open reproduction of deepseek-r1.

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*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, page 611–626, New York, NY, USA. Association for Computing Machinery.

Hynek Kydlíček. Math-Verify: Math Verification Library.

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. 2023. Taco: Topics in algorithmic code generation dataset. *arXiv preprint arXiv:2312.14852*.

Zihan Liu, Yang Chen, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. Acemath: Advancing frontier math reasoning with post-training and reward modeling. *arXiv preprint*.

Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. 2025. Reasoning models can be effective without thinking. *Preprint*, arXiv:2504.09858.

Anqi Mao, Mehryar Mohri, and Yutao Zhong. 2023. Cross-entropy loss functions: Theoretical analysis and applications. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 23803–23828. PMLR.

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. *Preprint*, arXiv:2501.19393.

Young-Jin Park, Kristjan Greenewald, Kaveh Alim, Hao Wang, and Navid Azizan. 2025. Know what you don't know: Uncertainty calibration of process reward models. *Preprint*, arXiv:2506.09338.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '20. IEEE Press.

ByteDance Seed, :, Jiaze Chen, Tiantian Fan, Xin Liu, Lingjun Liu, Zhiqi Lin, Mingxuan Wang, Chengyi Wang, Xiangpeng Wei, Wenyuan Xu, Yufeng Yuan, Yu Yue, Lin Yan, Qiying Yu, Xiaochen Zuo, Chi Zhang, Ruofei Zhu, Zhecheng An, and 255 others. 2025. Seed1.5-thinking: Advancing superb reasoning models with reinforcement learning. *Preprint*, arXiv:2504.13914.

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

Shuaijie She, Junxiao Liu, Yifeng Liu, Jiajun Chen, Xin Huang, and Shujian Huang. 2025. R-prm: Reasoning-driven process reward modeling. *Preprint*, arXiv:2503.21295.

Lin Sun, Chuang Liu, Xiaofeng Ma, Tao Yang, Weijia Lu, and Ning Wu. 2025. Freeprm: Training process reward models without ground truth process labels. *Preprint*, arXiv:2506.03570.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024a. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11897–11916, Bangkok, Thailand. Association for Computational Linguistics.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024b. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439.

Xinming Wang, Jian Xu, Aslan H Feng, Yi Chen, Haiyang Guo, Fei Zhu, Yuanqi Shao, Minsi Ren, Hongzhu Yi, Sheng Lian, and 1 others. 2025a. The hitchhiker's guide to autonomous research: A survey of scientific agents. *Authorea Preprints*.

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

Zixiao Wang, Yuxin Wang, Xiaorui Wang, Mengting Xing, Jie Gao, Jianjun Xu, Guangcan Liu, Chenhui Jin, Zhuo Wang, Shengzhuo Zhang, and Hongtao Xie. 2025b. Test-time scaling with reflective generative model. *Preprint*, arXiv:2507.01951.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Jianshu Chen Yunkai Zhang Hong Wang Shiyang Li Xiyou Zhou Wenhu Chen, Hongmin Wang and William Yang Wang. 2020. Tabfact : A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Shijie Xia, Xuefeng Li, Yixin Liu, Tongshuang Wu, and Pengfei Liu. 2024. Evaluating mathematical reasoning beyond accuracy. *arXiv preprint arXiv:2404.05692*.

Shijie Xia, Yiwei Qin, Xuefeng Li, Yan Ma, Run-Ze Fan, Steffi Chern, Haoyang Zou, Fan Zhou, Xiangkun Hu, Jiahe Jin, Yanheng He, Yixin Ye, Yixiu Liu, and Pengfei Liu. 2025. Generative ai act ii: Test time scaling drives cognition engineering. *Preprint*, arXiv:2504.13828.

Bin Yu, Hang Yuan, Haotian Li, Xueyin Xu, Yuliang Wei, Bailing Wang, Weizhen Qi, and Kai Chen. 2025a. Long-short chain-of-thought mixture supervised fine-tuning eliciting efficient reasoning in large language models. *Preprint*, arXiv:2505.03469.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, and 16 others. 2025b. Dapo: An open-source llm reinforcement learning system at scale. *Preprint*, arXiv:2503.14476.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. 2025. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *Preprint*, arXiv:2503.18892.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025a. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

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

Jian Zhao, Runze Liu, Kaiyan Zhang, Zhimu Zhou, Junqi Gao, Dong Li, Jiafei Lyu, Zhouyi Qian, Biqing Qi, Xiu Li, and Bowen Zhou. 2025a. Genprm: Scaling test-time compute of process reward models via generative reasoning. *arXiv preprint arXiv:2504.00891*.

Wenting Zhao, Pranjal Aggarwal, Swarnadeep Saha, Asli Celikyilmaz, Jason Weston, and Ilia Kulikov. 2025b. The majority is not always right: Rl training for solution aggregation. *Preprint*, arXiv:2509.06870.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2024. Processbench: Identifying process errors in mathematical reasoning. *arXiv preprint arXiv:2412.06559*.

Tong Zheng, Hongming Zhang, Wenhao Yu, Xiaoyang Wang, Runpeng Dai, Rui Liu, Huiwen Bao, Chengsong Huang, Heng Huang, and Dong Yu. 2025. Parallel-r1: Towards parallel thinking via reinforcement learning. *Preprint*, arXiv:2509.07980.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Jiaru Zou, Ling Yang, Jingwen Gu, Jiahao Qiu, Ke Shen, Jingrui He, and Mengdi Wang. 2025. Reasonflux-prm: Trajectory-aware prms for long chain-of-thought reasoning in llms. *Preprint*, arXiv:2506.18896.

A Evaluation Prompt Template

The prompt template used is sourced from LightEval (Habib et al., 2023).

Evaluation Prompt Template

Solve the following math problem efficiently and clearly. Please reason step by step, and put your final answer within \boxed{}.

B Training Dataset Statistics

The size of the training dataset is shown in the Table 4. As can be seen, the ratio of positive to negative samples we selected is maintained at 1:1.

Data Source	Positive	Negative	Total
DeepMath-103K	11,592	11,592	23,184
OpenR1-Math-220k	55,258	55,258	110,516
Overall	**66,850**	**66,850**	**133,700**

Table 4: Training Dataset Size

C Sampling Parameters

Table 5 presents the sampling parameters used during the policy model's inference time in the dataset construction process and benchmark evaluation. These parameter values are all derived from the official Best Practices recommended by Qwen3-8B (Team, 2025). The inference service for the LLM is deployed using vLLM (Kwon et al., 2023) as the foundation infrastructure.

Parameter	Value
Enable Thinking	True
Temperature	0.6
Top-p	0.95
Max tokens	10,000
Top-k	20
Min-p	0

Table 5: Sampling Parameters

```
class TrajSelector(PreTrainedModel):
  # this is the annotation
  def __init__(self, config):
    ...
    self.policy_model = load(model_name="Qwen3-8B")
    self.prm = load(model_name="Qwen3-0.6B-Base")
    self.projection = nn.Linear(
      policy_hidden_states_dim,
      prm_hidden_states_dim
    )
    self.score_head = nn.Sequential(
      nn.Linear(prm_hidden_states_dim, d),
      nn.ReLU(),
      nn.Linear(d, num_labels),
    )
    ...
```

Figure 5: PyTorch style code of TrajSelector.

D Training Parameters

Table 6 presents the training hyper-parameters used in the training process.

Parameter	Value
Samples	133K
Trainable Part	Full model
Learning Rate	1×10^{-4}
Epoch	3
Optimizer	AdamW
DeepSpeed	Zero2
Weight Decay	0.1
Max Seq.Length	10,000 tokens
Per-device Batch Size	2
Gradient Accumulation	4
Max tokens	10,000
Mixed Precision	bfloat16
GPU Nums	8× NVIDIA H100

Table 6: Training Parameters

E Pseudo Code of Network Architecture

Figure 5 shows the PyTorch implementation pseudo code for the TrajSelector network architecture.

F Offline Data Selection

We used OpenThoughts-114K (Guha et al., 2025) as the source dataset, from which 1K samples were selected as the supervised fine-tuning (SFT) dataset via different data selection strategies. Subsequently, we performed SFT on the Qwen2.5-14B-Instruct (Team, 2024) model; the performance of the fine-

tuned model was evaluated to assess the effectiveness of the various data selection strategies. Additionally, we employed two datasets for comparison: s1K (Muennighoff et al., 2025) (a carefully human-curated dataset) and a randomly selected dataset. All training strategies and parameter selections follow the settings in the s1 (Muennighoff et al., 2025), and the evaluation methods strictly adhere to the default strategy of the open-r1 (Hugging Face, 2025) evaluation suite. The rationale for this approach is that these experimental protocols have been explored in recent related studies (Zou et al., 2025; Yu et al., 2025a). The experimental results are presented in Table 7.

Dataset	MATH500	AIME24	AIME25	GPQA
random	71.6	16.7	20.0	34.8
s1K	78.8	40.0	33.3	41.4
Math-Shepherd-PRM-7B	67.8	13.3	6.7	33.3
Qwen2.5-Math-PRM-7B	73.2	26.7	20.0	39.4
ReasonFlux-PRM-7B	84.8	40.0	33.3	47.5
TrajSelector (our)	**86.4**	**43.3**	**43.3**	**53.5**

Table 7: Offline Data Selection Evaluation

As observed from the experimental results, the training samples selected by TrajSelector are more effective in improving model performance, which reflects TrajSelector's capability to assess the quality of thinking trajectories.

G Application in Reinforcement Learning

We apply TrajSelector as a lightweight process reward model in LLM reinforcement learning. Specifically, we employ Qwen3-8B as the policy model and modify the GRPO algorithm (Guo et al., 2025a) to integrate the original outcome-based reward (r_{out}) with the process-level rewards ($\hat{r}$) generated by our method, yielding a more fine-grained token-level reward signal:

$$r_t = \beta \cdot \hat{r} + (1 - \beta) \cdot r_{\text{out}}, \tag{6}$$

where $\hat{r}$ denotes the process reward at token position t produced by TrajSelector, r_{out} represents the outcome reward from GRPO, and $\beta = 0.5$ is a weighting hyperparameter that balances the two reward components.

G.1 Experimental Setup

We initialize the policy model from Qwen3-8B and employ GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021b) as training data. The model is trained using the modified GRPO (Shao et al., 2024) and DAPO (Yu et al., 2025b) algorithm with our hybrid reward function. Other experimental settings follow those in SimpleRL-Zoo (Zeng et al., 2025).

G.2 Results

Table 8 presents the performance comparison between the baselines and our modified version integrated with TrajSelector. We evaluate the models on four mathematical reasoning benchmarks: AIME24, AIME25, BeyondAIME, and BRUMO25. The evaluation metric is the average accuracy over 32 runs, denoted as $avg@32$.

Table 8: Performance comparison on reinforcement learning benchmarks.

Method	AIME24	AIME25	BeyondAIME	BRUMO25
Qwen3-8B	64.4	58.4	38.0	49.4
GRPO	77.6	64.2	39.0	56.9
DAPO	79.0	66.5	40.6	58.4
GRPO + **ours**	**80.2**	**67.8**	**41.5**	**59.4**
DAPO + **ours**	**82.3**	**68.7**	**42.8**	**60.8**

The results demonstrate that incorporating fine-grained process rewards from TrajSelector into the reinforcement learning algorithm leads to improved performance, addressing the reward sparsity issue inherent in outcome-only reinforcement learning approaches.

H Ablation Study on Sampler Model

To explore the feasibility of using models with more diverse scales as frozen sampler models, we conducted experimental evaluations using Qwen3-4B and Qwen3-14B respectively as the models for response generation. The experimental results are presented in the Table 9. From the experimental results, the adoption of TrajSelector still yields benefits compared to Majority Voting method.

I Larger N in Best-of-N

The experimental results for Best-of-48 and Best-of-64 are presented in the Table 10. The evaluation setup is consistent with Section 5.1 in the main experiment.

From the experimental results, TrajSelector maintains stable performance and continues to yield benefits when N takes larger values, demonstrating a scaling trend where its effectiveness improves as N increases.

Method	AMC-23 (/40)	AIME-24 (/30)	AIME-25 (/30)	BeyondAIME (/100)	HMMT-25 (/30)	BRUMO-25 (/30)	Avg (%)
Best-of-32 + Qwen3-4B as Sampler							
Pass@32 (Oracle)	**39**	**24**	**21**	**50**	**17**	**22**	**71.25**
Majority Voting	**38**	21	16	26	12	17	56.83
TrajSelector + 0.6B PRM (ours)	**38**	**22**	**17**	**30**	**12**	**18**	**59.17**
TrajSelector + 1.7B PRM (ours)	**38**	**24**	**20**	**32**	**13**	**20**	**63.94**
TrajSelector + 4B PRM (ours)	**39**	**24**	**20**	**35**	**15**	**20**	**65.97**
Best-of-32 + Qwen3-14B as Sampler							
Pass@32 (Oracle)	**39**	**24**	**22**	**53**	**17**	**24**	**73.42**
Majority Voting	39	21	14	34	13	21	60.25
TrajSelector + 0.6B PRM (ours)	**39**	**23**	**18**	**33**	**13**	**19**	**62.31**
TrajSelector + 1.7B PRM (ours)	**39**	**23**	**19**	**35**	**15**	**20**	**64.86**
TrajSelector + 4B PRM (ours)	**39**	**24**	**21**	**39**	**16**	**22**	**68.86**

Table 9: Experimental Results of Best-of-32 on Qwen3-4B and Qwen3-14B

Method	AMC-23 (/40)	AIME-24 (/30)	AIME-25 (/30)	BeyondAIME (/100)	HMMT-25 (/30)	BRUMO-25 (/30)	Avg (%)
Best-of-48							
Pass@48 (Oracle)	**38**	**24**	**24**	**46**	**16**	**23**	**71.83**
Random Selection	32	17	13	19	7	14	44.83
Majority Voting	36	22	17	25	11	19	57.50
ReasonFlux-PRM-7B	36	18	15	21	8	16	50.17
Qwen2.5-Math-PRM-7B	35	19	15	23	7	16	50.08
Qwen2.5-Math-7B-PRM800K	36	18	15	20	9	16	50.56
ReasonEval-7B	35	21	14	20	8	16	50.69
Math-Shepherd	34	11	14	23	7	13	43.00
AceMath-7B-RM	35	20	11	21	5	16	46.97
EurusPRM	36	20	17	27	12	18	56.72
TrajSelector + 0.6B PRM (our)	**38**	**22**	**19**	**33**	**12**	**19**	**61.33**
TrajSelector + 1.7B PRM (our)	**38**	**23**	**20**	**36**	**13**	**20**	**64.06**
TrajSelector + 4B PRM (our)	**38**	**24**	**21**	**39**	**14**	**21**	**66.78**
Best-of-64							
Pass@64 (Oracle)	**39**	**25**	**24**	**55**	**17**	**23**	**74.86**
Random Selection	36	15	13	25	4	14	44.72
Majority Voting	37	23	17	35	13	19	61.25
ReasonFlux-PRM-7B	34	20	17	29	9	17	54.00
Qwen2.5-Math-PRM-7B	34	20	16	27	9	16	52.56
Qwen2.5-Math-7B-PRM800K	35	19	16	28	9	16	52.58
ReasonEval-7B	35	18	15	22	7	16	49.36
Math-Shepherd	33	17	17	22	7	15	48.53
AceMath-7B-RM	33	16	14	20	5	18	46.52
EurusPRM	37	21	18	30	11	19	58.75
TrajSelector + 0.6B PRM (our)	**39**	**23**	**19**	**37**	**12**	**20**	**63.52**
TrajSelector + 1.7B PRM (our)	**39**	**24**	**21**	**40**	**14**	**21**	**67.36**
TrajSelector + 4B PRM (our)	**39**	**25**	**22**	**43**	**15**	**22**	**70.08**

Table 10: Experimental Results of Best-of-48 & Best-of-64

J Generalization to Code Generation and Tabular Reasoning

To examine whether TrajSelector is limited to mathematical reasoning, we additionally evaluate it on two strong reasoning domains: code generation and tabular reasoning. These domains require different forms of reasoning from olympiad-style mathematics: code generation emphasizes executable program synthesis and functional correctness, while tabular reasoning requires structured evidence aggregation over tabular inputs. The goal of this appendix is to report the detailed experimental settings and results for these cross-domain evaluations.

J.1 Code Generation

Experimental setting. We evaluate whether TrajSelector generalizes to code generation, where a correct solution is often not unique and must be judged by functional behavior rather than exact answer matching. We use APPS (Hendrycks et al., 2021a) and TACO (Li et al., 2023) as the training datasets, and evaluate on three standard code generation benchmarks: HumanEval (Chen et al., 2021a), MBPP (Austin et al., 2021), and LiveCodeBench v5 (Jain et al., 2024). Qwen3-8B is used as the backbone sampler model, and the sampling parameters follow the mathematical reasoning experiments unless otherwise specified. For each APPS/TACO training problem, Qwen3-8B generates one candidate program. We execute the generated program against the available unit tests and assign a binary outcome label according to functional correctness. Programs that fail any unit test, exceed the timeout or memory limit, or trigger runtime/compilation exceptions are treated as incorrect. We then balance positive and negative samples to construct the training set. Since our step segmentation procedure is task-agnostic, we use the same segmentation strategy as in mathematical reasoning: reasoning traces are split by double newlines, and step-final hidden states are used for process scoring. During evaluation, the sampler generates $N = 16$ candidate programs for each problem, and each selection method chooses one program as the final submission.

Baselines. Since semantically equivalent programs can differ substantially in surface form, majority voting over generated answers is not directly applicable to code generation. We therefore compare TrajSelector with two selection baselines. **Random Selection** uniformly samples one program from the 16 candidates. **LLM-as-a-Judge Ranking** uses Qwen3-32B as a judge model to score and rank each generated program, then selects the highest-ranked candidate. We report two variants of our method, TrajSelector with a 0.6B process verifier and TrajSelector with a 4B process verifier.

LLM-as-a-Judge protocol. For the LLM-as-a-Judge baseline, we prompt Qwen3-32B to evaluate each generated program independently rather than ranking all 16 candidates in a single prompt. This design reduces position bias and encourages the judge to assess functional correctness instead of relative surface quality. The prompt provides the problem statement, input-output specification, examples, and one candidate program, and instructs the judge to focus on whether the program is likely to pass hidden tests. The judge is asked to consider algorithmic correctness, edge cases, time and memory constraints, and implementation reliability, while ignoring readability, style, comments, and naming unless they affect correctness. It then returns a scalar score from 0 to 100 together with a short verdict. We rank the 16 candidate programs by this score and select the highest-scoring program as the final output.

Results. Table 11 shows that TrajSelector consistently improves Best-of-16 selection for code generation. With the lightweight 0.6B verifier, TrajSelector achieves an average score of 78.34, outperforming both Random Selection and the Qwen3-32B LLM-as-a-Judge baseline. Scaling the verifier to 4B further improves the average score to 80.10 and yields the best result on all three benchmarks. These results suggest that latent-state process scoring remains effective when correctness is defined by executable program behavior rather than a single canonical final answer.

J.2 Tabular Reasoning

Experimental setting. We further evaluate whether TrajSelector generalizes to tabular reasoning, where models must aggregate evidence from structured tables and, in some cases, combine tabular evidence with textual context. We use the training sets of FinQA (Chen et al., 2021b), TAT-QA (Zhu et al., 2021), and TabFact (Wenhu Chen and Wang, 2020) as training data. For each training problem, Qwen3-8B generates one candidate response using the same sampling parameters as the mathematical reasoning experiments. We then

Method	HumanEval	MBPP	LiveCodeBench	Avg
Random Selection	83.54	76.40	57.27	72.40
LLM-as-a-Judge Ranking	91.46	79.20	60.11	76.92
TrajSelector (0.6B PRM)	92.68	80.20	62.15	78.34
TrajSelector (4B PRM)	**95.12**	**81.20**	**63.98**	**80.10**

Table 11: Best-of-16 code generation results on HumanEval, MBPP, and LiveCodeBench v5. Results are reported as percentages, and higher values indicate better performance. All methods use Qwen3-8B as the backbone sampler model.

compare the extracted final answer with the ground-truth answer using task-specific answer checking to assign a binary outcome label. Responses with incorrect answers, unparsable final answers, or execution/parsing exceptions during evaluation are treated as incorrect, and positive and negative samples are balanced to construct the training set. Since step segmentation is task-agnostic, we use the same double-newline segmentation strategy and step-final hidden states as in mathematical reasoning and code generation. For evaluation, we use the test sets of FinQA, TAT-QA, and WikiTableQuestions (Pasupat and Liang, 2015). Following the code generation setup, Qwen3-8B is used as the backbone sampler model and each method is evaluated under the Best-of-16 setting.

Baselines. We compare TrajSelector with three baselines. **Qwen3-8B** denotes the direct single-response result from the backbone sampler. **Random** uniformly selects one response from the 16 candidates. **Majority Voting** normalizes the final answers and selects the answer that appears most frequently among the candidates. We report two variants of our method, TrajSelector with a 0.6B process verifier and TrajSelector with a 4B process verifier.

Results. Table 12 shows that TrajSelector improves Best-of-16 selection across all three tabular reasoning benchmarks. The 0.6B verifier achieves an average score of 68.98, outperforming both Random selection and Majority Voting. The 4B verifier further improves the average score to 69.27 and obtains the best result on FinQA, TAT-QA, and WikiTableQuestions. These results indicate that latent-state process scoring also benefits tabular reasoning, where successful selection requires identifying responses with reliable evidence aggregation over structured inputs.

K Generalization to More Model Families

To address concerns regarding the generalizability of our architecture across different model families, we conducted two supplementary experiments following the training and evaluation protocols described in this paper. Specifically, we evaluate TrajSelector on Best-of-64 selection using two additional sampler models:

- **DeepSeek-R1-Distill-Qwen-7B**: A 7B model based on the Qwen2.5 architecture.
- **DeepSeek-R1-Distill-Llama-8B**: An 8B model based on the Llama3 architecture.

These experiments demonstrate that TrajSelector is not constrained to specific model architectures, tokenizers, or pre-trained corpora, and consistently achieves superior performance compared to baseline methods.

K.1 Results on DeepSeek-R1-Distill-Qwen-7B

Table 13 presents the Best-of-64 results using DeepSeek-R1-Distill-Qwen-7B as the sampler model. Our method achieves the best performance across all five benchmarks, substantially outperforming both Majority Voting and existing 7B-scale PRMs.

K.2 Results on DeepSeek-R1-Distill-Llama-8B

Table 14 shows the Best-of-64 results using DeepSeek-R1-Distill-Llama-8B as the sampler model. Results are reported as accuracy percentages. Despite the different architecture (Llama3 vs. Qwen2.5), TrajSelector maintains its effectiveness and achieves the best performance across all benchmarks.

K.3 Discussion

The experimental results demonstrate several key findings:

Architecture agnostic. TrajSelector successfully generalizes across different model architectures (Qwen2.5 and Llama3), indicating that our approach is not tied to specific architectural choices or tokenization schemes.

Method	**FinQA**	**TAT-QA**	**WikiTableQuestions**	**Avg**
Qwen3-8B	26.59	90.23	83.29	66.70
Random	26.42	89.93	83.20	66.52
Majority Voting	30.69	91.01	84.12	68.61
TrajSelector (0.6B PRM)	31.56	91.13	84.27	68.98
TrajSelector (4B PRM)	**31.91**	**91.49**	**84.42**	**69.27**

Table 12: Best-of-16 tabular reasoning results on FinQA, TAT-QA, and WikiTableQuestions. Results are reported as percentages, and higher values indicate better performance. All Best-of-16 methods use Qwen3-8B as the backbone sampler model.

Method	AIME24	AIME25	BeyondAIME	HMMT25	BRUMO25
+ Majority Voting	83.3	70.0	48.0	46.7	73.3
+ ReasonFlux-PRM-7B	66.7	53.3	25.0	23.3	53.3
+ Qwen2.5-Math-PRM-7B	63.3	50.0	29.0	30.0	46.7
+ Qwen2.5-Math-7B-PRM800K	63.3	53.3	28.7	29.0	50.0
+ TrajSelector + 0.6B PRM (ours)	**90.0**	**76.7**	**51.0**	**53.3**	**73.3**
+ TrajSelector + 1.7B PRM (ours)	**93.3**	**80.0**	**54.0**	**56.7**	**76.7**
+ TrajSelector + 4B PRM (ours)	**96.7**	**83.3**	**57.0**	**60.0**	**80.0**

Table 13: Best-of-64 results on DeepSeek-R1-Distill-Qwen-7B across mathematical reasoning benchmarks.

Method	AIME24	AIME25	BeyondAIME	HMMT25	BRUMO25
+ Majority Voting	80.0	63.3	45.0	46.7	63.3
+ ReasonFlux-PRM-7B	56.7	46.7	22.0	20.0	46.7
+ Qwen2.5-Math-PRM-7B	53.3	43.3	22.0	23.3	50.0
+ Qwen2.5-Math-7B-PRM800K	56.7	43.3	23.0	26.7	50.0
+ TrajSelector + 0.6B PRM (ours)	**86.7**	**73.3**	**48.0**	**46.7**	**67.7**
+ TrajSelector + 1.7B PRM (ours)	**90.0**	**76.7**	**51.0**	**50.0**	**70.0**
+ TrajSelector + 4B PRM (ours)	**93.3**	**80.0**	**54.0**	**53.3**	**73.3**

Table 14: Best-of-64 results on DeepSeek-R1-Distill-Llama-8B across mathematical reasoning benchmarks.

Consistent superiority. Across both model families and all five benchmarks, TrajSelector consistently outperforms all baseline methods, including the widely-adopted Majority Voting and multiple strong 7B-scale PRMs.