Accelerating LLM Reasoning via Early Rejection with Partial Reward Modeling

Seyyed Saeid Cheshmi[1*] **Azal Ahmad Khan**[1*]
Xinran Wang[1] **Zirui Liu**[1] **Ali Anwar**[1]
[1]University of Minnesota
{chesh014, khan1069, wang8740, zrliu, aanwar}@umn.edu

Abstract

Large Language Models (LLMs) are increasingly relied upon for solving complex reasoning tasks in domains such as mathematics, logic, and multi-step question answering. A growing line of work seeks to improve reasoning quality by scaling inference time compute particularly through Process Reward Models (PRMs), used to reward the reasoning at intermediate steps. While effective, these methods introduce substantial computational overhead, especially when generating large numbers of solutions in parallel. In this paper, we investigate whether PRMs can be used mid-generation to provide early signals that enable the rejection of suboptimal candidates before full generation of step is complete. We introduce the hypothesis that PRMs are also Partial Reward Models, meaning that the scores they assign to partially completed reasoning step are predictive of final output quality. This allows for principled early rejection based on intermediate token-level signals. We support this hypothesis both theoretically, by proving that the risk of discarding optimal beams decreases exponentially with generation length and empirically, by demonstrating a strong correlation between partial and final rewards across multiple reward models. On math reasoning benchmarks, our method achieves up to 1.4×–9× reduction in inference FLOPs without degrading final performance. These results suggest that early rejection is a powerful mechanism for improving the compute-efficiency of reasoning in LLMs. The code and implementation are available at https://github.com/scheshmi/accelerated-reasoning-ER-PRM.

1 Introduction

Large Language Models (LLMs) are at the forefront of AI capabilities due to their emerging ability to perform complex reasoning tasks (Kojima et al., 2022; Chan, 2024; Cheng et al., 2025; Hazra et al., 2025; Xu et al., 2025). They have demonstrated significant success in domains such as mathematical problem solving, multi-hop question answering, and logical inference (Creswell et al., 2022; Ahn et al., 2024; Akella, 2024). These advancements are critical because they signal a shift from surface-level pattern recognition to deeper, multi-step deductive reasoning (Wei et al., 2022; Zhou et al., 2022). Enhancing these reasoning abilities is paramount for developing more capable, reliable models that can operate across various domains.

Prior Works. As the scaling of model parameters and pretraining data has started to become a bottleneck, recent efforts have shifted toward increasing compute at inference time to improve the reasoning capabilities of LLMs (Snell et al., 2024). Improving the reasoning capabilities of LLMs by scaling compute at inference time has been pursued through multiple strategies. A prominent approach leverages *Outcome Reward Models*, which train a separate evaluator to score the final output of the LLM based on correctness or quality (Cobbe et al., 2021; Hosseini et al., 2024; Mahan et al., 2024; Zhang et al., 2024b). Another approach uses, *Process Reward Models* (PRMs) to evaluate intermediate steps or reasoning trajectories generated during inference (Wang et al., 2023; Snell et al., 2024; Zhang et al., 2024a; Luo et al., 2024). Under this paradigm, the model generates multiple candidate reasoning paths, which are then evaluated by the PRM that assigns rewards at the end of each step. This step-wise evaluation enables the selection of promising trajectories for further expansion while allowing the rejection of less promising ones, thereby guiding the reasoning process more efficiently. Techniques such as beam search, Monte Carlo Tree Search (MCTS) guided by value models, and PRM-guided methods, exemplify this strategy (Feng et al., 2023; Yao et al., 2023). In this

* Equal contributions (ordered via coin-flip).

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 10433–10447
November 4-9, 2025

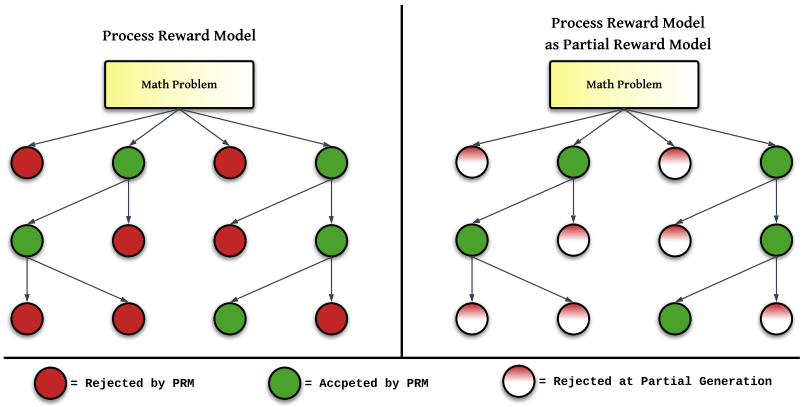

Figure 1: **Process-Reward Model (PRM) at full length vs. PRM reused for early rejection. (Left)** Every beam is expanded to full depth before the PRM scores its complete solution, so all intermediate branches incur compute even if they are were to fail. **(Right)** The same PRM is invoked mid-generation after each block of few tokens, producing a partial reward that serves as an early-quality signal.

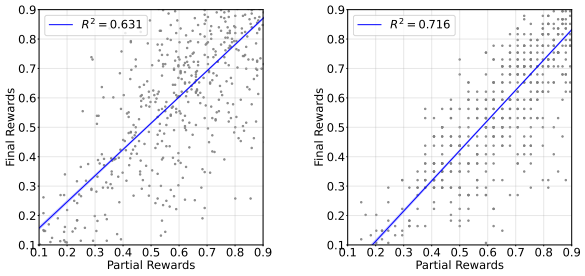

Figure 2: Linear relationship between partial rewards (reward calculated at half step completion) and full rewards (rewards calculated at full step completion) with **(left)** Llemma-MetaMath-7b and **(right)** MathShepherd-Mistral-7b as reward models.

paper, we focus specifically on the PRM paradigm and explore how to improve its efficiency.

Challenges. While scaling test-time compute using methods like PRMs can significantly enhance performance, it also introduces substantial computational overhead, especially for long sequences where many generated beams contribute little value (Chen et al., 2024; Hu et al., 2025; Wang et al., 2025). To be competitive with state-of-the-art post-training approaches, the number of beams often needs to be scaled up to 1000–60000, resulting in a large number of output tokens (Sun et al., 2024). These output tokens are not only computationally expensive to generate, but are also produced sequentially, leading to considerable latency (Yang et al., 2024). A natural solution is to *reject unpromising candidates early in the decoding process, after only a few tokens, before committing to full step generation*, a strategy we refer to as *Early Rejection*.However, a major challenge in early rejection is making sure that decisions based on only part of the output don't accidentally discard high-quality completions. This is difficult because the overall quality of a reasoning trace often depends on its full structure, which might not be obvious from the first few tokens. As a result, developing a reliable method that can make early yet accurate decisions about which traces to keep, based on partial generations, remains an important and open research problem.

Hypothesis. To address this challenge, we present the a hypothesis, ***Process Reward Models are also Partial Reward Models.*** That is, for structured reasoning tasks, the partial scores assigned by a PRM, when evaluated after a small but meaningful fraction of the generation, are sufficiently correlated with the final scores. This insight suggests that PRMs, which are conventionally applied at the end of a complete reasoning trace, can also be used mid-generation to provide partial rewards that act as early indicators of output quality. Figure 1 illustrates this distinction: while traditional PRM usage scores complete reasoning paths only at the end, our approach invokes the same PRM mid-generation to score partial traces, enabling early rejection of unpromising candidates. In doing so, they enable principled early rejection based on intermediate token-level signals. Preliminary results, as shown in Figure 2, reveal a consistent relationship between partial and final rewards, modeled as a monotonic mapping with added noise.

Contributions. This paper makes the following key contributions: **(C1)** *We introduce the hypoth-*

esis that Process Reward Models (PRMs) can be used as Partial Reward Models to enable early rejection of suboptimal beams. We support this hypothesis by showing that partial rewards computed after only a fraction of the generation are strongly correlated with final rewards, allowing for reliable early decisions in the reasoning process. **(C2)** *We provide theoretical guarantees that justify the use of partial scores for early rejection.* Specifically, we prove that under mild assumptions, the probability of prematurely rejecting the optimal trajectory decreases exponentially with the partial generation length. **(C3)** *We empirically demonstrate that early rejection guided by PRMs is both effective and compute-efficient.* On reasoning tasks such as AIME, Math-500 and AGI Eval, our approach reduces inference-time FLOPs by 1.4×–9× when using a mid-sized PRM (7B parameters) without any loss in task performance. Furthermore, when using a smaller PRM (1.5B parameters), we achieve up to 1.5×–4× reduction in FLOPs, demonstrating that even lightweight evaluators can enable highly efficient reasoning through early rejection.

2 Related Works

Generative Reward Models. Early approaches to guiding machine learning models relied on handcrafted heuristics, but as models have grown more complex, generative models have increasingly been used for supervision and alignment (Mahan et al., 2024). Generative models now serve as critics, verifiers, and, most notably, as reward models in RLHF (Mahan et al., 2024). Critics evaluate model outputs by providing detailed feedback (Luo et al., 2023; Lan et al., 2024; Lin et al., 2024; Du et al., 2024), while verifiers check the factual correctness or consistency of responses (Kouemo Ngassom et al., 2024; Qi et al., 2024; Kirchner et al., 2024). As reward models, they can be used to score either the final outcome (outcome reward models, ORMs) or provide feedback at intermediate steps (process reward models, PRMs) (Lightman et al., 2023a). ORMs deliver a single reward signal at the end of generation, while PRMs offer denser, stepwise supervision, which has been shown to improve reasoning and generalization (Cobbe et al., 2021; Wang et al., 2023; Hosseini et al., 2024; Zhang et al., 2024b; Snell et al., 2024; Luo et al., 2024). PRMs have also been shown to facilitate more interpretable learning dynamics by providing actionable feedback at each reasoning step, enabling finer-grained control over model behavior and accelerating convergence during training (Lightman et al., 2023a; Snell et al., 2024; Hosseini et al., 2024).

Early Rejection. In classification, confidence-based rejection and selective prediction methods (Geifman and El-Yaniv, 2019) allow models to withhold outputs for ambiguous or out-of-distribution inputs, while similar abstention strategies are used in regression (Mozannar and Sontag, 2020). In LLMs, early rejection began with Best-of-N (BoN) decoding, where all candidates are fully generated and only the best is selected (Cobbe et al., 2021; Zhou et al., 2022). Recent advances show that integrating PRMs as step-level re-rankers within beam search significantly boosts both accuracy and compute efficiency, as dense rewards allow for rejection of suboptimal reasoning paths and more effective exploration of diverse solutions (Wang et al., 2023; Snell et al., 2024; Zhang et al., 2024a; Luo et al., 2024). Speculative Rejection proposed using ORMs for early rejection in BoN by discarding weak candidates mid-generation (Sun et al., 2024). In this work, *we study the principle of early rejection for PRMs and demonstrate how it can be effectively integrated into beam search methods.*

3 Method

Beam Search for Reasoning. Beam search is a widely used decoding strategy in LLMs for structured generation tasks such as mathematical problem solving and multi-step reasoning (Yao et al., 2023; Feng et al., 2023; Snell et al., 2024). At each decoding step, the model expands a fixed-width set of N candidate beams by sampling multiple possible continuations and retaining only the top-scoring ones based on a predefined heuristic (e.g., log-probability or reward score). This iterative expansion and rejecting process allows the model to explore a larger space of possible outputs than greedy decoding, while remaining tractable compared to exhaustive search. In PRM-guided reasoning, each beam is scored at the end of every reasoning step by a PRM, which evaluates the coherence or correctness of the generated step. The highest scoring beams are then selected for further expansion, enabling the model to gradually construct a valid multi-step reasoning trace.

3.1 Partial Scoring for Early Rejection

Standard inference-time reasoning with LLMs and PRMs involves generating multiple candidate reasoning trajectories, typically using beam search or tree-based strategies, and scoring each trajectory after every step generation. Based on these step-wise scores, a subset of beams is selected and expanded further. While this strategy has been instrumental in advancing long-horizon reasoning, it incurs substantial computational overhead, as all candidate steps must be fully generated before evaluation, regardless of their quality.

We introduce a modification to this pipeline by reusing the same PRM mid-step generation. A compact overview is shown in Algorithm 3, where instead of waiting for a full step to complete, we compute partial rewards after first block of τ tokens at each step. These intermediate scores serve as early indicators of downstream quality. Beams with low partial scores are rejected before completing the full step. The surviving beams are then completed to the end of the current step, after which expansion proceeds as in the standard pipeline. Early rejection is applied again at the next step. This process ensures that computation is focused on the most promising candidates, reducing the number of unnecessary tokens generated and minimizing redundant PRM evaluations. A full version of the algorithm, along with the standard PRM-guided baseline, is provided in Appendix A for reproducibility and implementation details.

Algorithm 1 Beam Search with Early Rejection

1: Initialize N beams
2: **for** each beam **do**
3: Generate up to τ tokens and compute partial reward using PRM
4: **end for**
5: Select top N/M beams by partial reward and complete remaining beams to full step
6: Expand each remaining beam with M new beams
7: Repeat scoring, early rejection, and expansion until stopping condition is met
8: **return** Best final sequence

Figure 3: Overview of beam search with early rejection.

3.2 Efficiency Gains from Early Rejection

This early rejection strategy is focused on reduction in the number of tokens generated. By rejecting weaker candidates after a partial generation, we avoid expending compute on beams unlikely to contribute to the final output. The impact of this optimization on both generation cost and reward model evaluation is summarized below:

Early rejection reduces compute

Rejecting beams after generating first τ tokens leads to FLOPs reduction for each step generation and during PRM evaluation.

Beyond reducing total compute, early rejection also improves throughput through a two-tiered batching strategy. Since rejected beams only require τ tokens to be generated, they occupy significantly less memory. This enables to increase the batch size during the initial generation phase without getting OOM error. We then switch to a smaller batch size for completing the remaining beams, balancing exploration with memory efficiency. This batching decoupling is summarized below:

Two-tiered batching improves throughput

We use a larger batch size for generating the first τ tokens, taking advantage of their lower memory cost, and a smaller batch size for completing the step to avoid OOM error.

4 Theoretical Guarantees

Background and Notation. At each decoding step, which we define as a block of τ tokens, a width of N beams is maintained. For beam i, let P_i denote its *partial* reward after the first τ tokens and F_i its *final* reward after completing the step. Our preliminary results in Figure 2 indicate that the final reward is related to the partial reward via a monotonic mapping with added noise:

$$F_i = g(P_i) + \eta_i$$

where g: $[0, 1] \to [0, 1]$ is a monotonic increasing function; which need not be linear and η_i is a noise term with zero mean and variance σ^2 that can cause deviations from a perfect linear relationship. After the PRM assigns partial rewards, we keep only the top $\frac{N}{M}$ beams and expand each of them into M new beams, restoring the total width N. Let $p = \frac{N}{M}$, the selection threshold T is the $(1 - 1/M)$ quantile of the partial-reward distribution (i.e., we keep the

top N/M beams). Therefore, a beam survives only if $P_i \geq T$.

Let the beam that would eventually yield the highest final score be

$$i^* \;=\; \arg\max_{i \in N} F_i$$

Guarantee under noisy, nonlinear conditions. Although the mapping between P_i and F_i need not be linear, we assume (i) the noise terms η_i are independent and σ-sub-Gaussian, and (ii) the expected partial scores preserve the ordering of the expected final scores. Let

$$\Delta \;=\; \min_{j \neq i^*} \big(\mathbb{E}[P_{i^*}] - \mathbb{E}[P_j]\big) \;>\; 0$$

denote the smallest expected gap between the best beam i^* and any other beam. Thus

$$\begin{aligned}\Pr\big(P_{i^*} < T\big) &\leq \Pr\Big(\exists\, j \neq i^* :\; P_j > P_{i^*}\Big) \\ &\leq (N-1)\,\exp\Big(-\tfrac{\Delta^2}{4\sigma^2}\Big),\end{aligned}$$

where the last step applies a sub-Gaussian tail bound to each pairwise difference $P_{i^*} - P_j$ and then takes a union bound over the $N-1$ non-optimal beams. The bound decays *exponentially* in Δ^2/σ^2; thus, when the expected gap is appreciable and the noise is modest, the risk of pruning the optimal beam is negligible even for large beam widths.

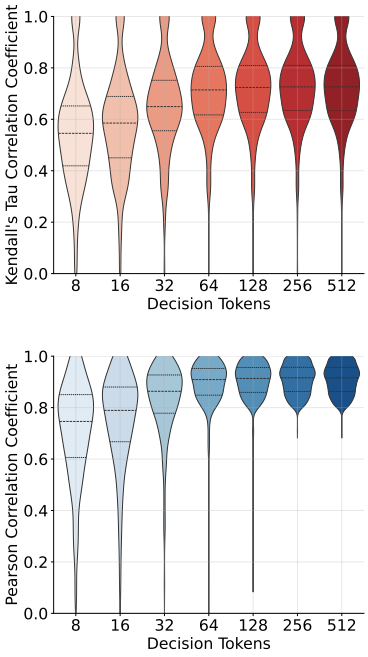

Figure 4: **(Top)** Kendall's Tau and **(Bottom)** Pearson's correlation coefficient for the partial and final rewards.

Best τ for Early Rejection. A common toy model is to treat each token's (log-)score as an i.i.d. random variable. For beam i, let $X_{i,1}, \ldots, X_{i,L}$ be i.i.d. with mean μ_i and variance σ_i^2, where L denotes the final sequence length (number of tokens at completion) and $1 \leq \tau \leq L$. The partial reward after τ tokens is $P_i = \sum_{t=1}^{\tau} X_{i,t}$, while the final reward is $F_i = \sum_{t=1}^{L} X_{i,t}$. Under this model the Pearson correlation reads

$$\rho(P_i, F_i) = \frac{\mathrm{Cov}(P_i, F_i)}{\sqrt{\mathrm{Var}(P_i)}\sqrt{\mathrm{Var}(F_i)}} \;=\; \sqrt{\frac{\tau}{L}}.$$

The shared first τ tokens drive the entire covariance: as $\tau \to L$ the correlation approaches 1 meaning the partial score is an almost perfect proxy, whereas as $\tau \to 0$ it vanishes. Figure 4 shows that this $\sqrt{\tau/L}$ trend, tightening toward 1 as τ increases, is also true empirically.

If we require the correlation to exceed a target level ρ^*, then

$$\rho(P_i, F_i) = \sqrt{\tfrac{\tau}{L}} \;\geq\; \rho^* \quad\Longrightarrow\quad \tau \;\geq\; (\rho^*)^2 L.$$

For example, attaining $\rho^* = 0.8$ demands $\tau \geq 0.64\,L$.

Connection to the Sub-Gaussian Bound. Our rejection guarantee hinges on

$$\Pr\big(P_{i^*} < T\big) \;\leq\; (N-1)\,\exp\Big(-\tfrac{\Delta^2}{4\sigma^2}\Big),$$

where $\Delta = \min_{j \neq i^*} \big(\mathbb{E}[P_{i^*}] - \mathbb{E}[P_j]\big)$ is the expected partial-score gap and σ is the sub-Gaussian parameter of the per-token noise.

A high correlation $\rho(P_i, F_i)$ does not automatically imply a large gap Δ, but it does indicate that beams ranking highly under the partial reward tend to rank highly under the final reward. In practice, choosing τ so that

$$\rho(P_i, F_i) = \sqrt{\tfrac{\tau}{L}} \;\geq\; \rho^*$$

ensures the partial scores are sufficiently predictive; once this condition is met, the tail bound above tells us the probability of mistakenly pruning the optimal beam is exponentially small in Δ^2/σ^2. In practice, after fixing τ we measure the empirical gap Δ on a held-out set and confirm it comfortably exceeds the estimated noise scale σ.

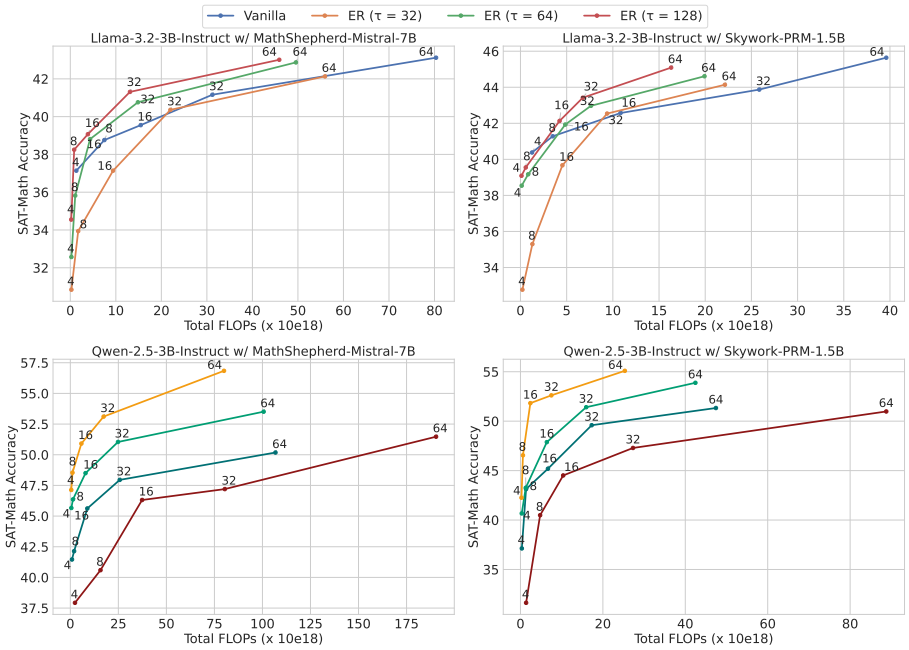

Figure 5: We evaluate our implementation of Early Rejection (ER) on the SAT-MATH dataset from AGIEval benchmark using two different LLMs and PRMs. The numbers indicate that ER rejection achieves performance similar to Vanilla Beam Search with while consuming far less compute.

5 Experiments

We evaluate our method on three challenging math-reasoning benchmarks, MATH-500 (Lightman et al., 2023b), SAT-MATH from AGIEval (Zhong et al., 2023), and AIME 2024. For generation we use the instruct variants of two open-source LLMs, Llama-3.2-3B (Meta, 2024) and Qwen-2.5-3B (Qwen et al., 2024), selected for their strong reasoning ability at modest scale. Process evaluation is performed with two PRMs of different capacities, MathShepherd-Mistral-7B (Wang et al., 2023) and Skywork-PRM-1.5B, allowing us to study the impact of early rejection for PRMs of different sizes. Early rejection is triggered after a prefix of $\tau \in 32, 64, 128$ tokens. These thresholds are motivated by preliminary analysis (Figure 4), which shows that partial-reward scores at these lengths are already highly correlated with final rewards. At each decoding step we sample $N \in 4, 8, 16, 32, 64$ candidate beams and retain the top $M = 4$, mirroring prior PRM-guided search settings (Snell et al., 2024). We compare our early-rejection decoder with the conventional pipeline that scores only fully completed beams, reporting average answer accuracy and total inference FLOPs for each run. All experiments are conducted on an HPC cluster, with each run executed using four NVIDIA A100 GPUs (40 GB memory each).

5.1 Experimental Results

Experimental results in Figure 5 on SAT-MATH dataset and 6 on Math-500 and AIME 2024 datasets highlight the effectiveness of Early Rejection (ER) in reducing compute while preserving end-task accuracy across different PRMs, LLMs, and τ values. For the results we observe that across all configurations, early rejection acts as a safe and compute-efficient strategy that adapts well to LLM characteristics and PRM granularity. Appendix A provides a comprehensive breakdown of accuracy and compute trade-offs across all datasets, τ values, beam sizes, and LLM–PRM configurations. Building on these results, we articulate five key observations that our subsequent experiments directly address.

Observation ❶: Partial PRM scores at very short prefixes reliably predict final scores. Our empirical analysis confirms that partial rewards become highly predictive of final rewards after surprisingly short prefixes. Figure 4 shows as we sweep the decision prefix τ from 8 to 512 tokens. The two correlations rise monotonically and follow the $\sqrt{\tau/L}$ and at $\tau = 32$ tokens ρ already exceeds 0.78 and $\tau = 64$ pushes both metrics above 0.9, after which they plateau. A complementary view is given in Figure 2, where a linear fit between half-step partial rewards and full-step rewards achieves $R^2 = 0.63$ with the MetaMath-7B PRM and $R^2 =$

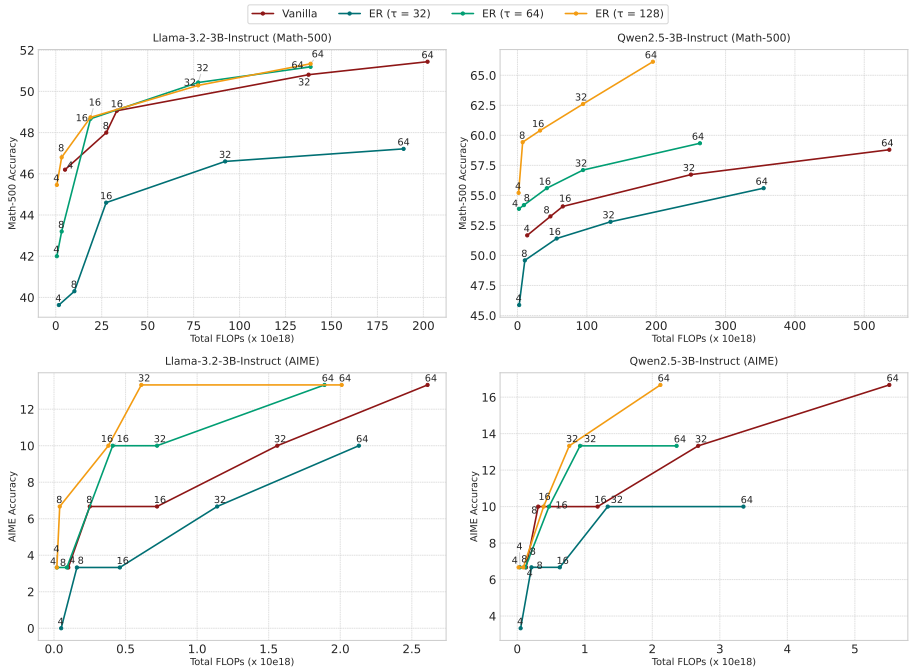

Figure 6: We evaluate our implementation of Early Rejection (ER) on the Math-500 and AIME datasets using two different LLMs with MathShepard-7b as reward model. The numbers indicate that ER rejection achieves performance similar to Vanilla Beam Search with while consuming far less compute.

0.72 with MathShepherd-7B, demonstrating that the effect generalizes across reward models . These findings validate our Partial Reward Model hypothesis that even a one-third length prefix offers a stable ranking signal, and the probability of incorrectly rejecting the optimal beam decays exponentially once the expected partial-score gap Δ dominates the sub-Gaussian noise σ, as formalized in Section 4. Practically, this means we can invoke early rejection after the first 32–64 tokens with negligible risk while removing 60–85% of downstream PRM calls and generation FLOPs.

Observation ❷: Smaller PRMs can match or exceed larger PRMs in accuracy while saving more compute, especially on well-structured outputs. The smaller Skywork-PRM-1.5B achieves equal or higher end-task accuracy than the MathShepherd-Mistral-7B baseline, while also enabling a higher number of FLOP reductions. Across both Llama-3.3-3B and Qwen2.5-3B, Skywork yields a 0.7–2.1% accuracy gain for smaller beam sizes and stays within 0.3% elsewhere, contradicting the common intuition that *"bigger judge = better answers"* (Leike et al., 2018). We also observe a greater number of FLOP reductions with Skywork-PRM-1.5B, primarily because the 3B-sized LLM becomes the computational bottleneck, and early rejection allows us to skip costly completions, thereby saving compute more frequently.

Another key observation is that smaller PRMs benefit from more well-structured answers. Skywork-PRM-1.5B generally performs better with Llama-3.3-3B than with Qwen2.5-3B, as Llama tends to produce more structured and instruction-following responses compared to Qwen. Although both LLMs are instruction-tuned, Llama adheres to instructions more faithfully, making it easier for the smaller PRM (Skywork) to evaluate intermediate steps accurately. In contrast, larger PRMs like MathShepherd-Mistral-7B are more robust to such variations in LLM behavior.

Observation ❸: Early rejection yields large accuracy gains for exploratory LLMs at small beam widths but offers diminishing accuracy returns for deterministic LLMs and wider beams. Qwen2.5-3B often generates long, exploratory reasoning traces, so many beams appear weak after the first $\tau = 32$–64 tokens, even though some of them would eventually reach correct solutions. In such cases, the partial reward filter discards the clearly unpromising beams early. Here early rejection frees up beam slots for new candidates. This allows the search to explore a broader set of reasoning paths, effectively expanding the search space without increasing the beam width N.

In contrast, Llama-3.2-3B tends to produce

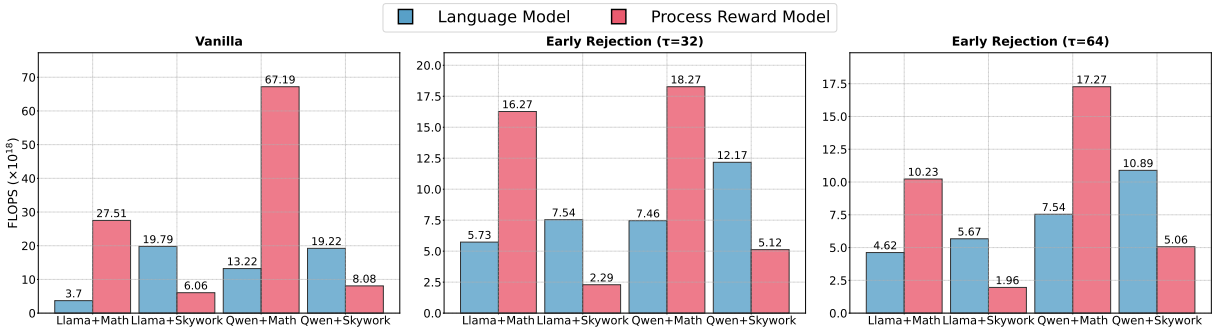

Figure 7: Total FLOPs consumed across different LLM–PRM combinations with and without Early Rejection. We observe consistent and substantial reductions in compute, with $\tau = 64$ yielding up to 9× savings. Larger prefix lengths enable more reliable pruning, significantly lowering overall inference cost without compromising accuracy.

shorter, more deterministic traces where the top-p beams already rank highly from the start. As a result, early rejection removes fewer low-quality candidates and provides limited additional exploration. Empirically, early rejection improves Qwen's accuracy by up to 3.5% at $N = 4$ and 1.6% at $N = 8$, whereas Llama sees at most a 0.3% gain. Once the beam width is sufficiently large ($N \geq 32$), the baseline search already explores the space well, so the benefits of early rejection shift from accuracy gains to compute savings.

Observation ❹: At $\tau = 64$ tokens, early rejection achieves higher accuracy while lesser compute than $\tau = 32$ tokens. Although we always retain the same number of beams per step (the top N/M), their quality improves significantly as we increase the prefix length τ. At $\tau = 32$, the correlation between partial and full rewards is about 0.78. This means around 20% of the beams are incorrectly ranked, so some low-quality beams make it through and have to be fully generated and evaluated, wasting compute. At $\tau = 64$, the correlation exceeds 0.90 and flattens out, meaning nearly all retained beams are genuinely promising. Very few low-quality beams slip through. As a result, even though we keep the same number of survivors, the number of bad survivors and the FLOPs spent on them, drops when increasing τ from 32 to 64.

Observation ❺: Language model behavior (not size) drives compute, and early rejection is most effective when it blocks exploratory failures early. Figure 7 shows that Qwen2.5-3B incurs significantly higher total FLOPs than Llama-3.2-3B under identical early rejection settings. While both models are similar in size, their generation behaviors differ: Qwen tends to produce longer, exploratory chains of thought, whereas Llama generates more concise, deterministic outputs. As a result, when early rejection fails to prune a weak Qwen beam, it often leads to a long and costly completion, inflating total compute. Early rejection is most effective in these exploratory settings, where catching bad completions early prevents large downstream FLOPs. This explains why Qwen exhibits larger absolute FLOP reductions, especially when paired with a lightweight PRM like Skywork-1.5B. In contrast, Llama's beams tend to converge quickly, offering fewer opportunities for savings. These results highlight that the structure of the generation process, not just model size, governs the impact of early rejection on efficiency.

6 Conclusion

We demonstrate that PRMs can be effectively repurposed as Partial Reward Models, enabling a single mid-generation evaluation to provide a reliable accept or reject signal. This allows weak beams to be pruned early, well before full reasoning steps are completed, thereby reducing unnecessary computation without sacrificing final accuracy. Under mild noise assumptions, we provide theoretical guarantees showing that the probability of mistakenly discarding the optimal beam decays exponentially with prefix length, offering formal safety for early rejection. Extensive experiments across SAT-MATH, Math-500, and AIME confirm the practical benefits: early rejection reduces inference-time FLOPs by 1.4×–9× when using a mid-sized PRM (7B parameters), with no degradation in task performance. Even with a smaller PRM (1.5B), we observe 1.5×–4× compute savings, highlighting that lightweight evaluators are sufficient for effective and efficient reasoning. Together, these findings establish early rejection as a simple, model-agnostic plug-in that narrows the gap between compute-

heavy tree search and fast single-pass decoding, offering state-of-the-art compute efficiency without compromising solution quality.

Limitations

Our approach relies on the monotonicity and calibration of PRM scores, if partial rewards correlate weakly with final quality, as might occur in tasks with delayed or non-monotonic utilities (e.g., code synthesis with backtracking or creative writing), early rejection can mis-reject the eventual best beam. The study is confined to text-only, math-centric benchmarks. Larger models specially for multimodal tasks, or domains with sparse positive signals may exhibit different trade-offs. While we report FLOP reductions, we do not quantify the memory overhead of storing intermediate PRM states after τ tokens are generated. Finally, the theoretical guarantees assume independent step-wise noise and fixed τ, leaving open questions about adaptive τ schedules and integration with policy-learning frameworks such as RLHF or DPO.

Ethical Considerations

While Early Rejection reduces inference compute by up to $9\times$, this efficiency could also facilitate misuse, such as the automated generation of spam or disinformation. The method's safety relies on the assumption that PRM scores are monotonic with respect to final output quality. However, this assumption may not hold beyond the math-focused benchmarks evaluated in this work, particularly for tasks involving delayed or non-monotonic rewards. As a result, the algorithm risks discarding high-quality candidates prematurely or reinforcing hidden biases.

Acknowledgments

The work of Azal Ahmad Khan was supported in part by the Amazon Machine Learning Systems Fellowship and the UMN GAGE Fellowship. Xinran Wang and Ali Anwar were supported by the 3M Science and Technology Graduate Fellowship and the Samsung Global Research Outreach Award.

References

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157*.

Amogh Akella. 2024. Improving math problem solving in large language models through categorization and strategy tailoring. *arXiv preprint arXiv:2411.00042*.

Emunah Chan. 2024. Understanding logical reasoning ability of large language models. *Available at SSRN 4943448*.

Zhaorun Chen, Zhuokai Zhao, Zhihong Zhu, Ruiqi Zhang, Xiang Li, Bhiksha Raj, and Huaxiu Yao. 2024. Autoprm: Automating procedural supervision for multi-step reasoning via controllable question decomposition. *arXiv preprint arXiv:2402.11452*.

Fengxiang Cheng, Haoxuan Li, Fenrong Liu, Robert van Rooij, Kun Zhang, and Zhouchen Lin. 2025. Empowering llms with logical reasoning: A comprehensive survey. *arXiv preprint arXiv:2502.15652*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Antonia Creswell, Murray Shanahan, and Irina Higgins. 2022. Selection-inference: Exploiting large language models for interpretable logical reasoning. *arXiv preprint arXiv:2205.09712*.

Jiangshu Du, Yibo Wang, Wenting Zhao, Zhongfen Deng, Shuaiqi Liu, Renze Lou, Henry Peng Zou, Pranav Narayanan Venkit, Nan Zhang, Mukund Srinath, and 1 others. 2024. Llms assist nlp researchers: Critique paper (meta-) reviewing. *arXiv preprint arXiv:2406.16253*.

Xidong Feng, Ziyu Wan, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. 2023. Alphazero-like tree-search can guide large language model decoding and training. *arXiv preprint arXiv:2309.17179*.

Yonatan Geifman and Ran El-Yaniv. 2019. SelectiveNet: A deep neural network with an integrated reject option. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2151–2159. PMLR.

Rishi Hazra, Gabriele Venturato, Pedro Zuidberg Dos Martires, and Luc De Raedt. 2025. Have large language models learned to reason? a characterization via 3-sat phase transition. *arXiv preprint arXiv:2504.03930*.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. 2024. V-star: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*.

Pengfei Hu, Zhenrong Zhang, Qikai Chang, Shuhang Liu, Jiefeng Ma, Jun Du, Jianshu Zhang, Quan Liu, Jianqing Gao, Feng Ma, and 1 others. 2025. Prm-bas: Enhancing multimodal reasoning through

prm-guided beam annealing search. *arXiv preprint arXiv:2504.10222*.

Jan Hendrik Kirchner, Yining Chen, Harri Edwards, Jan Leike, Nat McAleese, and Yuri Burda. 2024. Prover-verifier games improve legibility of llm outputs. *arXiv preprint arXiv:2407.13692*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Sylvain Kouemo Ngassom, Arghavan Moradi Dakhel, Florian Tambon, and Foutse Khomh. 2024. Chain of targeted verification questions to improve the reliability of code generated by llms. In *Proceedings of the 1st ACM International Conference on AI-Powered Software*, pages 122–130.

Tian Lan, Wenwei Zhang, Chen Xu, Heyan Huang, Dahua Lin, Kai Chen, and Xian-Ling Mao. 2024. Criticeval: Evaluating large-scale language model as critic. *Advances in Neural Information Processing Systems*, 37:66907–66960.

Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. 2018. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023a. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

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

Zicheng Lin, Zhibin Gou, Tian Liang, Ruilin Luo, Haowei Liu, and Yujiu Yang. 2024. Criticbench: Benchmarking llms for critique-correct reasoning. *arXiv preprint arXiv:2402.14809*.

Liangchen Luo, Zi Lin, Yinxiao Liu, Lei Shu, Yun Zhu, Jingbo Shang, and Lei Meng. 2023. Critique ability of large language models. *arXiv preprint arXiv:2310.04815*.

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*.

Dakota Mahan, Duy Van Phung, Rafael Rafailov, Chase Blagden, Nathan Lile, Louis Castricato, Jan-Philipp Fränken, Chelsea Finn, and Alon Albalak. 2024. Generative reward models. *arXiv preprint arXiv:2410.12832*.

Meta. 2024. Llama 3.2: Revolutionizing edge ai and vision with open, customizable models.

Hussein Mozannar and David Sontag. 2020. Consistent estimators for learning to defer to an expert. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7076–7087. PMLR.

Jianing Qi, Hao Tang, and Zhigang Zhu. 2024. Verifierq: Enhancing llm test time compute with q-learning-based verifiers. *arXiv preprint arXiv:2410.08048*.

Team Qwen, Baosong Yang, B Zhang, B Hui, B Zheng, B Yu, Chengpeng Li, D Liu, F Huang, H Wei, and 1 others. 2024. Qwen2 technical report. *arXiv preprint*.

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*.

Hanshi Sun, Momin Haider, Ruiqi Zhang, Huitao Yang, Jiahao Qiu, Ming Yin, Mengdi Wang, Peter Bartlett, and Andrea Zanette. 2024. Fast best-of-n decoding via speculative rejection. *arXiv preprint arXiv:2410.20290*.

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*.

Teng Wang, Zhangyi Jiang, Zhenqi He, Wenhan Yang, Yanan Zheng, Zeyu Li, Zifan He, Shenyang Tong, and Hailei Gong. 2025. Towards hierarchical multi-step reward models for enhanced reasoning in large language models. *arXiv preprint arXiv:2503.13551*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, and 1 others. 2025. Towards large reasoning models: A survey of reinforced reasoning with large language models. *arXiv preprint arXiv:2501.09686*.

Yuqing Yang, Yuedong Xu, and Lei Jiao. 2024. A queueing theoretic perspective on low-latency llm inference with variable token length. *arXiv preprint arXiv:2407.05347*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

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

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. 2024b. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2023. Agieval: A human-centric benchmark for evaluating foundation models. *arXiv preprint arXiv:2304.06364*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and 1 others. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

A Appendix

Algorithm 2 Beam Search with Process Reward Models

1: **Input:** LLM Model, PRM Model, Beam count N, Beam width M, Temperature T, Stopping criterion, EOS token or max search depth, Batch Size b
2: Initialize a set of N candidate beams
3: **for** each beam **do**
4: Sample N independent steps using the LLM with temperature T
5: Apply the stopping criterion (e.g., new line or double new line)
6: **end for**
7: Score each sampled step using the PRM
8: Select the top N/M steps with the highest scores
9: Expand the selected steps:
10: **for** each selected step **do**
11: Sample M next steps
12: **end for**
13: **while** EOS token not reached **and** max search depth not exceeded **do**
14: Repeat steps 7 - 12
15: **end while**
16: **return** The best sequence found

Algorithm. Algorithm 2 shows the conventional PRM-guided beam search and Algorithm 3 shows our early-rejection variant. Both algorithms maintain the same top-level structure of iterative beam expansion, but differ critically in how and when PRM scores are computed. The standard method evaluates only fully completed beams, resulting in redundant computation on unpromising candidates. In contrast, our early-rejection variant computes partial rewards after just τ tokens using the same PRM, enabling efficient early rejecting. This architectural shift introduces a two-tiered batching scheme, larger batch size for partial generations and smaller batch size for step completion, yielding significant compute savings without degrading performance, as shown in our experimental results.

Results. To supplement the main results presented in Section 5, we provide detailed tables reporting the accuracy and compute trade-offs for every combination of language model (LLM), process reward model (PRM), beam size, and early rejection threshold τ. These results span three math reasoning benchmarks: SAT-MATH, Math-500, and AIME.

Table 1 reports the results on the SAT-MATH dataset from AGIEval. For each LLM–PRM pair, we compare standard decoding ("Vanilla") with our early rejection method across multiple τ values. Each cell reports both the accuracy and the total FLOPs used for inference. We observe that early rejection achieves similar or higher accuracy at significantly reduced compute, especially with exploratory LLMs like Qwen-2.5B.

Table 2 extends the analysis to the Math-500 and AIME 2024 benchmarks, using MathShepherd-Mistral-7B as the PRM. Again, we observe consistent trends across datasets: as τ increases, early rejection becomes more selective and cost-efficient, with only minor losses (if any) in final accuracy.

Table 3 aggregates FLOP consumption across all LLM–PRM combinations under three decoding regimes: Vanilla, ER(τ=32), and ER(τ=64). The results reveal that early rejection with $\tau = 64$ consistently achieves the lowest compute cost without compromising output quality, yielding up to 9× reduction in total inference FLOPs.

Together, these tables validate the scalability and robustness of our early rejection method across models, evaluators, datasets, and rejection thresholds.

Table 1: SAT-MATH results comparing vanilla decoding and Early Rejection (ER) across multiple beam sizes and τ values. Each cell reports (top) accuracy and (bottom) total inference FLOPs ($\times 10^{18}$).

Model	**PRM**	**Setting**	**Number of Samples (τ)**				
			4	**8**	**16**	**32**	**64**
Llama-3.2 -3b	**MathSheperd -7b**	*Vanilla*	37.14 1.32	38.76 7.48	39.55 15.47	41.16 31.21	43.12 80.34
		ER (τ = 32)	30.84 0.24	33.94 2.73	35.14 9.40	40.36 21.99	42.13 55.94
		ER (τ = 64)	32.57 0.24	35.82 1.08	38.81 4.34	40.76 14.85	42.87 49.55
		ER (τ = 128)	34.55 0.21	38.25 0.85	39.07 3.86	38.31 13.11	40.65 45.90
	Skywork -1.5b	*Vanilla*	40.38 1.25	41.28 3.49	42.57 10.83	43.87 25.85	45.64 39.60
		ER (τ = 32)	32.77 0.21	35.30 1.29	39.67 4.54	38.54 9.38	44.14 22.13
		ER (τ = 64)	38.54 0.13	39.17 0.83	41.93 4.85	42.97 7.63	44.61 19.92
		ER (τ = 128)	32.24 0.11	33.33 0.57	37.21 4.23	39.09 6.75	39.55 16.31
Qwen2.5 -3b	**MathSheperd -7b**	*Vanilla*	37.93 2.42	40.59 15.70	46.31 37.35	47.20 80.41	51.47 190.35
		ER (τ = 32)	41.46 0.86	42.14 1.96	45.62 8.85	47.95 25.73	50.18 106.77
		ER (τ = 64)	45.66 0.53	46.36 1.37	48.50 7.91	51.04 24.81	53.51 100.61
		ER (τ = 128)	47.13 0.49	48.54 1.12	50.91 5.76	53.11 17.33	56.84 79.98
	Skywork -1.5b	*Vanilla*	31.63 1.37	40.49 4.77	44.51 10.37	47.29 27.31	50.98 88.77
		ER (τ = 32)	37.13 0.33	43.13 1.36	45.19 6.67	49.59 17.29	51.33 47.43
		ER (τ = 64)	40.67 0.31	43.26 1.28	47.88 6.40	51.41 15.95	53.88 42.45
		ER (τ = 128)	42.26 0.25	46.55 0.60	51.82 2.40	52.61 7.50	55.09 25.33

Table 2: Results on Math-500 and AIME datasets with MathShepherd-Mistral-7B as the PRM. Each configuration shows accuracy (top) and total FLOPs (bottom) for different beam sizes and τ thresholds.

Dataset	Model	Setting	Number of Samples (τ)				
			4	**8**	**16**	**32**	**64**
Math-500	**Llama-3.2 -3b**	*Vanilla*	46.20 5.04	48.00 27.51	49.06 33.22	50.81 137.54	51.44 202.27
		ER (τ = 32)	39.63 1.68	40.30 10.15	44.60 27.42	46.60 92.15	47.21 189.23
		ER (τ = 64)	42.00 1.50	43.20 8.67	48.67 23.45	50.43 101.17	51.19 184.71
		ER (τ = 128)	45.46 0.60	46.80 3.21	48.74 18.91	50.29 77.46	51.34 138.63
	Qwen2.5 -3b	*Vanilla*	51.67 14.02	53.25 47.48	54.08 65.32	56.73 250.03	58.80 536.10
		ER (τ = 32)	45.87 2.41	49.59 10.58	51.41 56.49	52.80 134.12	55.60 354.91
		ER (τ = 64)	53.88 2.10	54.19 9.28	55.60 42.33	57.11 112.46	59.34 263.08
		ER (τ = 128)	55.21 1.61	59.43 7.45	60.40 32.54	62.61 94.52	66.13 195.23
AIME	**Llama-3.2 -3b**	*Vanilla*	3.33 0.10	6.67 0.25	6.67 0.72	10.00 1.56	13.33 2.61
		ER (τ = 32)	0.00 0.05	3.33 0.16	3.33 0.46	6.67 1.14	10.00 2.13
		ER (τ = 64)	3.33 0.02	3.33 0.09	10.00 0.41	10.00 0.72	13.33 1.89
		ER (τ = 128)	3.33 0.02	6.67 0.04	10.00 0.38	13.33 0.61	13.33 2.01
	Qwen2.5 -3b	*Vanilla*	6.67 0.13	10.00 0.31	10.00 1.19	13.33 2.68	16.67 5.51
		ER (τ = 32)	3.33 0.05	6.67 0.21	6.67 0.63	10.00 1.34	10.00 3.35
		ER (τ = 64)	6.67 0.04	6.67 0.12	10.00 0.47	13.33 0.93	13.33 2.36
		ER (τ = 128)	6.67 0.02	6.67 0.09	10.00 0.39	13.33 0.77	16.67 2.12

Algorithm 3 Beam Search with Early Rejection

Input: LLM Model, PRM Model, Beam count N, Beam width M, Temperature T, Stopping criterion, EOS token or max search depth, $b_1 > b_2$
Initialize a set of N candidate beams
for each beam **do**
    Sample N independent steps using the LLM with temperature T and batch size b_1
    Apply the stopping criterion (τ tokens generated or EOS token.)
end for
Score each sampled step using the PRM
Select the top N/M steps with the highest scores
Complete the selected steps:
for each selected step **do**
    Complete the step until EOS token with batch size b_2.
end for
Expand the selected steps:
for each selected step **do**
    Sample M next steps
end for
while EOS token not reached **and** max search depth not exceeded **do**
    Repeat steps 7 - 16
end while
return The best sequence found

Table 3: Total FLOPs ($\times 10^{18}$) for each LLM–PRM combination under vanilla decoding and early rejection at $\tau = 32$ and $\tau = 64$. Early rejection consistently reduces compute, with Qwen-based configurations showing the largest savings.

Model Combination	**Vanilla**		**Early Rejection (τ=32)**		**Early Rejection (τ=64)**	
	LLM	**PRM**	**LLM**	**PRM**	**LLM**	**PRM**
Llama+Math	3.70	27.51	5.73	16.27	4.62	10.23
Llama+Skywork	19.79	6.06	7.54	2.29	5.67	1.96
Qwen+Math	13.22	67.19	7.46	18.27	7.54	17.27
Qwen+Skywork	19.22	8.08	12.17	5.12	10.89	5.06