# PROCESS REWARDS WITH LEARNED RELIABILITY

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Process Reward Models (PRMs) provide step-level feedback for reasoning, but current PRMs usually output only a single reward score for each step. Downstream methods must therefore treat imperfect step-level reward predictions as reliable decision signals, with no indication of when these predictions should be trusted. We propose BETAPRM, a distributional PRM that predicts both a step-level success probability and the reliability of that prediction. Given step-success supervision from Monte Carlo continuations, BETAPRM learns a Beta belief that explains the observed number of successful continuations through a Beta-Binomial likelihood, rather than regressing to the finite-sample success ratio as a point target. This learned reliability signal indicates when a step reward should be trusted, enabling downstream applications to distinguish reliable rewards from uncertain ones. As one application, we introduce Adaptive Computation Allocation (ACA) for PRM-guided Best-of-$N$ reasoning. ACA uses the learned reliability signal to stop when a high-reward solution is reliable and to spend additional computation on uncertain candidate prefixes. Experiments across four backbones and four reasoning benchmarks show that BETAPRM improves PRM-guided Best-of-$N$ selection while preserving standard step-level error detection. Built on this signal, ACA improves the accuracy–token tradeoff over fixed-budget Best-of-16, reducing token usage by up to $33.57\%$ while improving final-answer accuracy.

## 1 INTRODUCTION

Process Reward Models (PRMs) (Duan et al., 2025; He et al., 2024b; Li et al., 2026; Luo et al., 2024; Ma et al., 2023; Wang et al., 2024a;c; Zhang et al., 2025c; Zheng et al., 2025) provide step-level feedback for reasoning by scoring the intermediate steps of a solution. Because these step-level scores can guide candidate selection (Chae et al., 2026; Hu et al., 2025; Liu et al., 2025a) and policy optimization (Dai et al., 2024; Liu et al., 2025b), PRMs have become a useful interface for both test-time scaling (Chen et al., 2026; Kim et al., 2025) and reinforcement learning (Liu et al., 2026; Zhang et al., 2026). However, existing PRMs typically expose this interface as a single point estimate of step correctness, such as the probability that a step is correct. Downstream methods (Guan et al., 2025; Luo et al., 2025; Zhang et al., 2025b) often have to treat this imperfect score as a reliable decision signal, because no additional signal is available. A single PRM score tells us which step or candidate the model prefers, but not whether that preference should be trusted. As a result, an unreliable score can directly affect downstream decisions without being identified as uncertain.

As shown in Fig. 1, this classic interface mismatches both test-time usage and training supervision:

First, a single scalar reward cannot capture the predictive uncertainty of intermediate steps. At inference time, a causal PRM judges a step from the problem and current prefix, without seeing future continuations (Song et al., 2025; Tu et al., 2025). Even when no local error is obvious, it is uncertain whether a seemingly correct prefix will lead to a correct final answer. A more natural PRM output should capture both the estimated probability of success and the uncertainty of that estimate.

Second, step-level PRM labels are often noisy finite-sample estimates. A common source of supervision (Wang et al., 2024c; 2026; Xiong et al., 2025; Zhang et al., 2025c) samples $N$ continuations from a reasoning prefix and counts how many reach the correct final answer. If $K$ continuations succeed, the empirical ratio $K/N$ is only a Monte Carlo estimate of the prefix success probability, not the true underlying probability. Repeating the procedure from the same prefix could yield a different $K$ due to sampling randomness. Standard PRM training (Du et al., 2025; Duan et al., 2025; Li et al., 2026) nevertheless regresses to this observed ratio as a point label, forcing the model to fit



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

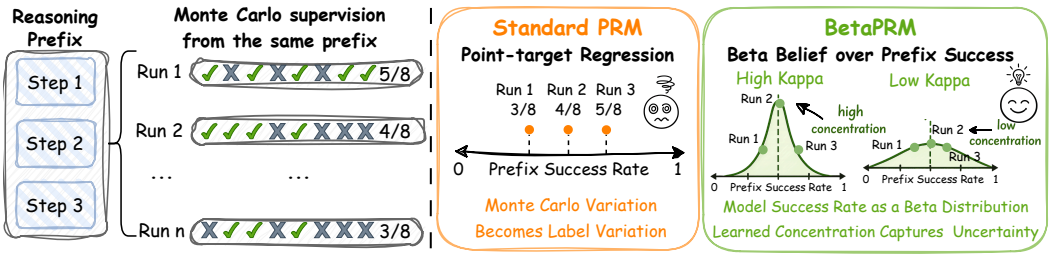


Figure 1: Motivation of BETAPRM. Repeated Monte Carlo continuations from the same prefix can produce different empirical success ratios. Standard PRMs treat these ratios as point targets, whereas BETAPRM models the prefix success probability as a Beta belief. The Beta mean $\mu$ gives the process reward, while the concentration $\kappa$ captures the reliability of the estimate, allowing the model to assign likelihood to the observed count $K$ out of $N$ rather than treating $K/N$ as an exact point label.

a noisy finite-sample outcome with a single scalar prediction. A better objective should keep the supervision in counting form: the model should assign high probability to observing $K$ successes out of $N$ continuations, rather than only regress to the single ratio $K/N$.

In this paper, we address both limitations by giving the PRM a way to express uncertainty about its own prediction. A step-level reward supported by a confident belief should not be treated the same as one produced under ambiguity. This motivates BETAPRM, a distributional PRM that predicts both how promising a reasoning prefix is and how reliable that prediction is. As illustrated in Figure 2, BETAPRM predicts a Beta distribution over the prefix success probability, and is trained so that this distribution can explain the Monte Carlo observations from sampled continuations. This distribution is parameterized by (1) **the predicted success probability** $\mu$, which serves as the usual PRM score, and (2) **the concentration** $\kappa$, which controls how tightly the belief is centered around that prediction. High concentration gives a sharp belief, while low concentration gives a flattened belief that can explain a wider range of Monte Carlo observations.

The learned concentration changes how PRM scores can be used. Rather than treating every scalar reward as equally trustworthy, downstream algorithms can distinguish confident rewards from uncertain ones. It is broadly useful for PRM-guided decision making; in this paper, we demonstrate one concrete test-time use case: Adaptive Computation Allocation (ACA) for Best-of-$N$ reasoning. Fixed-budget Best-of-$N$ (Lightman et al., 2024) spends the same rollout budget on every problem, even when the pool already contains a high-scoring candidate whose PRM judgment is reliable. ACA spends the budget through progressive batches: it stops when the selected answer is reliably ahead, and otherwise continues from uncertain prefixes where more computation may change the decision.

Empirically, BETAPRM improves PRM-guided Best-of-$N$ selection across four backbones and four benchmarks (*e.g.*, $+3.37$ points on average on InternVL2.5-8B), while preserving standard step-level error detection ability. Further analyses show that the learned concentration provides a nontrivial reliability signal. Built on this reliability signal, ACA improves the inference-time accuracy-token tradeoff compared with vanilla Best-of-16, where it reduces token usage by up to $33.57\%$ and even pushes final-answer accuracy higher.

## 2 RELATED WORK

**Process Reward Models.** PRMs (Du et al., 2025; Sun et al., 2025; Pala et al., 2025; Khalifa et al., 2025) provide step-level feedback for reasoning, unlike outcome reward models (Cobbe et al., 2021; Yu et al., 2024) that score only final answers. Prior work trains PRMs either as step judges for local error detection (Wang et al., 2026; Duan et al., 2025), or as Q-value-style models that estimate whether a prefix can be completed correctly (Li et al., 2026). We focus on a limitation of the latter view: Monte Carlo continuations provide finite-sample evidence about prefix success, yet existing methods often collapse this evidence into a single point label. We instead make reliability part of the PRM output, so downstream methods can use both the predicted reward and its reliability.

**Test-Time Scaling.** Test-time scaling (Huang et al., 2025; Snell et al., 2024; Brown et al., 2024; Uesato et al., 2022; You et al., 2025) improves reasoning by spending more inference compute,

including voting (Wang et al., 2023), verifier-guided selection (Zheng et al., 2026), and search over reasoning paths (Guan et al., 2025). A common and simple instance is Best-of-$N$ (Lightman et al., 2024): sample multiple candidate solutions and select one using a verifier or reward model. Most Best-of-$N$ methods use a fixed budget (Brown et al., 2024), allocating the same number of samples to every problem despite large variation in difficulty. Recent methods (Park et al., 2026) calibrate PRM success estimates to choose instance-specific budgets for sampling complete solutions. In contrast, our method uses BETAPRM's reward and learned reliability during generation to decide when to stop and which uncertain prefix to continue.

# 3 PRELIMINARIES

## 3.1 PREFIX-CONDITIONED PROCESS REWARDS

Given an input problem $x$, let $s_{1:T} = (s_1, \ldots, s_T)$ denote a step-by-step solution. We insert a special process marker <prm> after each step, and the PRM produces a score at each marker position:

$$x, s_1, \texttt{<prm>}, s_2, \texttt{<prm>}, \ldots, s_T, \texttt{<prm>}.$$

Since the reward model is a causal language model, the score at the $t$-th marker is computed from the prefix $c_t = (x, s_{\le t})$, without access to future steps $s_{t+1:T}$. This matches the online use of PRMs in generation or search, where a partial reasoning state is evaluated before its continuation is observed.

We therefore interpret process rewards as prefix-level quantities. Instead of assigning an isolated correctness label to step $t$, we define its quality as the prefix success probability $q_t = \Pr(\text{final answer is correct} \mid x, s_{\le t})$. Since $q_t$ is a latent variable, the next subsection describes how finite continuation samples provide supervision to learn this variable.

## 3.2 MONTE CARLO STEP SUPERVISION

The prefix success probability $q_t$ is an unobserved latent variable. A widely used way to construct step-level supervision is to sample $N$ continuations from a prefix $c_t = (x, s_{\le t})$ and count how many reach the correct final answer. Let $K_t$ denote the number of successful continuations. The empirical ratio $\hat{q}_t = K_t/N$ is a Monte Carlo estimate of $q_t$.

Standard PRM objectives (Du et al., 2025; Li et al., 2026; Wang et al., 2024c; 2025; Xiong et al., 2025; Zhang et al., 2025c) often reduce this observation to a single point target by optimizing cross-entropy against $\hat{q}_t$, $\mathcal{L}_{\text{CE}} = -\hat{q}_t \log p_t - (1 - \hat{q}_t) \log(1 - p_t)$, where $p_t$ is the predicted step score. This treats the empirical ratio as if it were the latent prefix success probability itself. Because $\hat{q}_t$ is computed from a small number of continuations, repeating the same procedure could produce a different $K_t$. Thus, forcing the model to learn the single point estimate $\hat{q}_t$ might lead to overfitting to sample noise. Instead, it is more natural to treat the supervision as a count observation ($K_t$ success out of $N$ trials).

# 4 BETAPRM

## 4.1 BETA-BINOMIAL COUNT MODEL

To formalize the count-based supervision, we assume a binomial generative process for the successful continuations: $K_t \mid q_t \sim \text{Binomial}(N, q_t)$. Because $q_t$ is an unknown latent success probability in $[0, 1]$, we model it with a Beta belief, $q_t \sim \text{Beta}(\alpha_t, \beta_t)$, which naturally pairs with the Binomial count observation above. For better interpretability, we reparameterize the Beta distribution by its mean $\mu_t = \alpha_t/(\alpha_t + \beta_t)$ and concentration $\kappa_t = \alpha_t + \beta_t$. Under this formulation, $\mu_t$ acts as the expected success probability (the standard PRM output score), while $\kappa_t$ controls how sharply the belief is concentrated around that mean. Marginalizing out the latent $q_t$ yields a Beta-Binomial distribution over $K_t$, providing a likelihood for count observations rather than a point target for $\hat{q}_t$.

## 4.2 BETAPRM PARAMETERIZATION

BETAPRM instantiates the Beta belief by predicting its mean and concentration at each process marker. At the $t$-th <prm> marker, the language model produces a hidden state $h_t$ and vocabulary

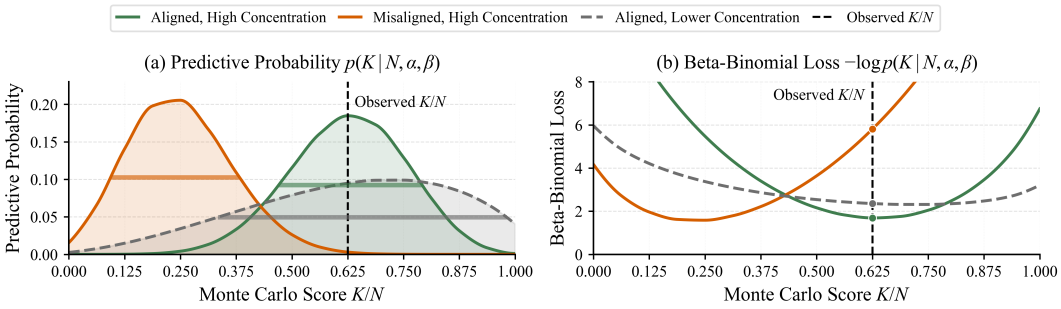


Figure 2: Intuition of Beta-Binomial supervision. A predicted Beta belief over prefix success induces a distribution over possible observed success ratios $K/N$. The green curve is concentrated and aligned with the observed count, the orange curve is concentrated but misaligned and thus penalized, and the gray curve has lower concentration and allows a wider range of finite-sample observations.

logits $z_t$. Let $z_t^{\text{Yes}}$ and $z_t^{\text{No}}$ denote the logits of the two reward tokens `Yes` and `No`. We define the predicted success probability by applying a softmax only over these two logits:

$$\mu_t = \frac{\exp(z_t^{\text{Yes}})}{\exp(z_t^{\text{Yes}}) + \exp(z_t^{\text{No}})}.$$

This preserves the standard PRM interpretation of the `Yes` probability as the scalar reward.

BETAPRM predicts a separate concentration parameter $\kappa_t = \text{softplus}(g_\phi(h_t)) + \kappa_{\min}$ to estimate reliability, where $g_\phi$ is a lightweight linear head and $\kappa_{\min}$ is a small fixed lower bound for numerical stability. This separates the reward from the reliability channel: the reward-token logits determine $\mu_t$, while the additional head determines how concentrated the model's belief should be.

The Beta parameters are then derived using $\alpha_t = \mu_t\kappa_t$ and $\beta_t = (1-\mu_t)\kappa_t$. Here $\mu_t$ centers the belief over prefix success and serves as the scalar PRM score, while $\kappa_t$ controls the concentration, allowing prefixes with similar scores to carry different reliability estimates.

### 4.3 BETA-BINOMIAL TRAINING OBJECTIVE

We train the predicted Beta belief by maximizing the likelihood of the observed count $K_t$. As shown in Figure 2, a concentrated belief centered near the observed ratio assigns high probability to the count, while a concentrated but misaligned belief receives a large loss. A lower-concentration belief spreads probability mass over a wider range of possible finite-sample observations, reflecting lower confidence.

Using the Beta-Binomial formulation, the predictive probability of the observed count is

$$p(K_t \mid N, \alpha_t, \beta_t) = \binom{N}{K_t} \frac{B(K_t + \alpha_t, N - K_t + \beta_t)}{B(\alpha_t, \beta_t)},$$

where $B(\cdot,\cdot)$ is the Beta function. Let $\mathcal{P}$ be the set of supervised process markers in a mini-batch. We define the Beta-Binomial loss, $\mathcal{L}_{\text{Beta-Binomial}}$, as the negative log-likelihood of the observed counts:

$$\mathcal{L}_{\text{Beta-Binomial}} = -\frac{1}{|\mathcal{P}|} \sum_{t \in \mathcal{P}} \log p(K_t \mid N, \alpha_t, \beta_t).$$

Minimizing this loss encourages the model to assign high probability to the observed count.

We add an auxiliary regularization loss to explicitly encourage calibrated reliability estimates. If $\mu_t$ disagrees with the observed ratio $K_t/N$, a large $\kappa_t$ indicating high confidence would be inconsistent with the evidence. We therefore penalize the product of disagreement and concentration, and add this regularizer to the Beta-Binomial loss:

$$\mathcal{L}_{\text{reg}} = \frac{\lambda_{\text{reg}}}{|\mathcal{P}|} \sum_{t \in \mathcal{P}} \left| \text{sg}(\mu_t) - \frac{K_t}{N} \right| \kappa_t, \qquad \mathcal{L} = \mathcal{L}_{\text{Beta-Binomial}} + \mathcal{L}_{\text{reg}}.$$

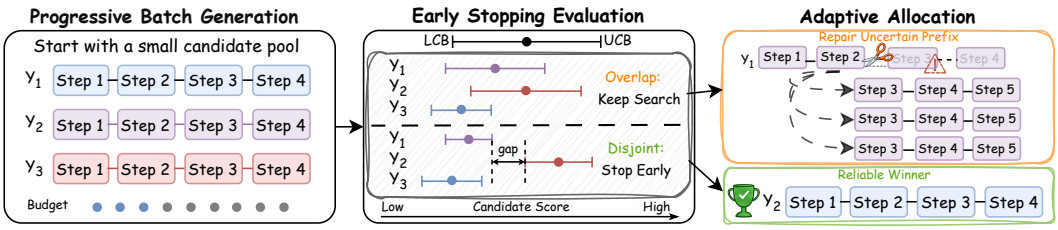


Figure 3: Overview of Adaptive Computation Allocation (ACA). ACA generates candidates in batches, uses BETAPRM scores and reliability estimates to test whether the current winner remains reliably ahead, and otherwise samples new continuations from uncertain prefixes.

Here, $\mathrm{sg}(\cdot)$ denotes the stop-gradient operation. The stop-gradient operation prevents this auxiliary term from directly pulling $\mu_t$ toward the noisy ratio, which would make it another point-label regression loss. Instead, the term penalizes high concentration when $\mu_t$ disagrees with the count evidence, with the penalty diminishing as they become consistent.

## 5 RELIABILITY-AWARE INFERENCE: ADAPTIVE COMPUTATION ALLOCATION

BETAPRM outputs both a reward mean and a reliability estimate. As shown in Figure 3, we study a straightforward inference-time use case: allocating computation in PRM-guided Best-of-$N$ reasoning. In standard practices (Cobbe et al., 2021), Best-of-$N$ improves inference by sampling multiple candidate solutions and selecting one according to a scoring rule, which can be a process reward model. In addition, every query receives the same number of sampled rollouts. We introduce Adaptive Computation Allocation (ACA) that saves computation when the sampled pool may already contain a high-scoring answer. ACA utilizes BETAPRM to estimate uncertainty and mainly follows two rules: (1) stop early when a reliable answer is found, and (2) redirect computation for uncertain prefixes.

**Risk-Adjusted Candidate Score.** ACA compares complete candidates using both reward and reliability. We quantify step-level uncertainty by the standard deviation of the predicted Beta distribution, $\sigma_t = \sqrt{\mu_t(1-\mu_t)/(\kappa_t+1)}$. For a fixed $\mu_t$, larger $\kappa_t$ yields smaller $\sigma_t$, indicating a more concentrated belief about the prefix success probability. For a candidate $y = s_{1:T}$, we define the risk-adjusted score $S(y) = T^{-1}\sum_{t=1}^{T}(\mu_t - \lambda\sigma_t)$, where $\lambda \geq 0$ controls the uncertainty penalty.

**Progressive Batch Generation and Early Stopping.** Standard Best-of-$N$ generates all $N$ candidates in one shot. ACA instead spends the budget in a progressive way: it first samples a small pool of $n_0$ candidates, scores them with BETAPRM, and then either stops or allocates another batch, up to the maximum budget $N$.

At each stage, ACA selects the highest-scoring candidate $y^\star = \arg\max_y S(y)$ for the stopping test, where we construct lower and upper confidence bounds (LCB and UCB):

$$\mathrm{LCB}(y) = S(y) - c_{\mathrm{stop}}U(y), \qquad \mathrm{UCB}(y) = S(y) + c_{\mathrm{stop}}U(y), \qquad U(y) = \frac{1}{T}\sum_{t=1}^{T}\sigma_t,$$

where $c_{\mathrm{stop}}$ scales the width of the confidence bounds. ACA terminates the allocation process for the current problem and returns $y^\star$ if $\mathrm{LCB}(y^\star) > \max_{y \neq y^\star} \mathrm{UCB}(y)$.

This criterion means that the highest-scoring candidate dominates the current pool: even its pessimistic score exceeds the optimistic score of every competitor. In this case, further expanding the pool with additional continuations is unlikely to change the PRM-guided selection.

**Uncertainty-Guided Prefix Repair.** If the stopping criterion is not met, ACA spends the next batch on a competitive existing response, chosen as the non-winner candidate with the highest UCB, where additional computation is most likely to change the current decision. To choose where to repair this response, ACA uses a deterministic cutpoint rule over reasoning steps. It first computes a conservative step score $\mu_t - c_{\mathrm{cut}}\sigma_t$ and selects the earliest step whose value falls below a low-quality threshold $p_{\mathrm{bad}}$. If no such step exists, ACA falls back to the most uncertain eligible reasoning step,

Table 1: PRM-guided Best-of-16 final-answer accuracy. All PRMs select from the same candidate pools generated by InternVL2.5-8B. Values with gray ↑ / ↓ indicate improvement/decline over the single-pass baseline, and Avg. Δ averages this improvement/decline over the four benchmarks.

| Selector | MathVision | OlympiadBench | MathVerse | MathVista | Avg. Δ |
|---|---|---|---|---|---|
| Single Pass | 18.08 | 8.65 | 35.31 | 52.77 | – |
| ***InternVL3-14B*** | | | | | |
| +Base as PRM | 19.74 ↑1.66 | 11.33 ↑2.68 | 36.17 ↑0.86 | 52.50 ↓0.27 | ↑1.23 |
| +Standard PRM | 23.03 ↑4.95 | **16.67** ↑8.02 | 45.41 ↑10.10 | 60.70 ↑7.93 | ↑7.75 |
| +BETAPRM | **25.66** ↑7.58 | **16.67** ↑8.02 | **46.35** ↑11.04 | **62.30** ↑9.53 | ↑**9.04 (+1.29)** |
| ***InternVL3-8B*** | | | | | |
| +Base as PRM | 18.75 ↑0.67 | 13.33 ↑4.68 | 37.21 ↑1.90 | 52.40 ↓0.37 | ↑1.72 |
| +Standard PRM | 22.69 ↑4.61 | 15.33 ↑6.68 | 44.80 ↑9.49 | 60.00 ↑7.23 | ↑7.00 |
| +BETAPRM | **24.34** ↑6.26 | **18.00** ↑9.35 | **45.20** ↑9.89 | **61.10** ↑8.33 | ↑**8.46 (+1.46)** |
| ***InternVL2.5-8B*** | | | | | |
| +Base as PRM | 20.72 ↑2.64 | 9.33 ↑0.68 | 36.83 ↑1.52 | 51.90 ↓0.87 | ↑0.99 |
| +Standard PRM | 21.38 ↑3.30 | 11.33 ↑2.68 | 42.81 ↑7.50 | 57.60 ↑4.83 | ↑4.58 |
| +BETAPRM | **25.66** ↑7.58 | **15.33** ↑6.68 | **44.31** ↑9.00 | **61.30** ↑8.53 | ↑**7.95 (+3.37)** |
| ***Qwen2.5-VL-7B*** | | | | | |
| +Base as PRM | 15.46 ↓2.62 | 8.00 ↓0.65 | 35.84 ↑0.53 | 50.70 ↓2.07 | ↓1.20 |
| +Standard PRM | 21.38 ↑3.30 | 14.00 ↑5.35 | 44.92 ↑9.61 | 60.30 ↑7.53 | ↑6.45 |
| +BETAPRM | **24.34** ↑6.26 | **17.33** ↑8.68 | **45.99** ↑10.68 | **63.60** ↑10.83 | ↑**9.11 (+2.66)** |

i.e., the step with the largest $\sigma_t$. The selected step is treated as a cutpoint: ACA keeps the prefix before the cutpoint, discards the subsequent generation, and samples new continuations from that prefix. The procedure repeats until the confidence condition holds or the budget $N$ is reached.

# 6 EXPERIMENTS

## 6.1 EXPERIMENTAL SETUP

We evaluate our proposed methods from two aspects. First, we evaluate BETAPRM as a PRM on PRM-guided Best-of-$N$ selection and step-level error detection. Second, we evaluate whether its uncertainty estimates improve Adaptive Computation Allocation (ACA) in Best-of-$N$ reasoning.

We use VisualPRM400K-v1.1[1] (Wang et al., 2026), which reports $K$ successful continuations out of $N$ Monte Carlo samples for each prefix. Before any model training, we uniformly sample 3,000 questions (16,553 prefixes) from this corpus to form a question-disjoint validation set for hyperparameter selection and reliability analysis, excluding all associated rollouts and annotated steps from training. The standard PRM baseline is trained with cross-entropy using the empirical ratio $K/N$ as a soft label, while BETAPRM uses the Beta-Binomial objective on $(K, N)$. We evaluate BETAPRM as a PRM with four backbones: InternVL2.5-8B (Chen et al., 2024b), InternVL3-8B (Zhu et al., 2025), InternVL3-14B (Zhu et al., 2025), and Qwen2.5-VL-7B (Bai et al., 2025). Best-of-$N$ selection uses candidate pools generated by InternVL2.5-8B (Chen et al., 2024b) and reports final-answer accuracy on MathVision (Wang et al., 2024b), OlympiadBench (He et al., 2024a), MathVerse (Zhang et al., 2024), and MathVista (Lu et al., 2024). Step-level error detection is evaluated on VisualProcessBench (Wang et al., 2026). ACA is evaluated on two representative backbones, InternVL2.5-8B (Chen et al., 2024b) and Qwen2.5-VL-7B (Bai et al., 2025), against fixed-budget Best-of-$N$ under the same maximum budget $N = 16$, reporting accuracy and generated tokens. Full training, evaluation, and ACA implementation details are provided in Appendix A.

## 6.2 BETAPRM EVALUATION

**BETAPRM improves Best-of-$N$ selection across four backbones and four benchmarks.** Table 1 evaluates PRMs as solution selectors under the same candidate pools. Standard PRM selects the

[1] https://huggingface.co/datasets/OpenGVLab/VisualPRM400K-v1.1-Raw

Table 2: Step-level error detection on VisualProcessBench. Overall denotes micro-F1 over all annotated steps, while each source column reports macro-F1 on that subset.

| Model | Overall | MathVision | MathVerse | MMMU | DynaMath | WeMath |
|---|---|---|---|---|---|---|
| *InternVL3-14B* | | | | | | |
| Base as PRM | 49.40 | 47.61 | 50.97 | 48.86 | 50.19 | 47.34 |
| Standard PRM | 61.90 | 59.90 | 61.93 | 63.12 | 62.93 | 63.79 |
| BETAPRM | 61.90 | 60.94 | 62.74 | 59.18 | 61.67 | 64.59 |
| *InternVL3-8B* | | | | | | |
| Base as PRM | 48.39 | 47.21 | 48.91 | 47.01 | 49.41 | 49.00 |
| Standard PRM | 60.69 | 60.20 | 60.41 | 58.73 | 61.63 | 63.23 |
| BETAPRM | 61.85 | 59.65 | 62.17 | 62.42 | 62.71 | 64.23 |
| *InternVL2.5-8B* | | | | | | |
| Base as PRM | 52.28 | 52.40 | 52.04 | 50.21 | 54.85 | 49.95 |
| Standard PRM | 61.54 | 60.78 | 60.47 | 62.05 | 62.91 | 64.38 |
| BETAPRM | 60.97 | 60.43 | 60.70 | 60.48 | 63.00 | 59.98 |
| *Qwen2.5-VL-7B* | | | | | | |
| Base as PRM | 49.68 | 50.22 | 49.58 | 49.85 | 49.62 | 48.51 |
| Standard PRM | 62.23 | 62.17 | 61.25 | 61.44 | 62.88 | 65.55 |
| BETAPRM | 62.91 | 62.19 | 62.91 | 59.49 | 63.75 | 66.69 |

candidate with the highest average process reward. BETAPRM exposes both a reward mean and a learned reliability estimate, so we use its full output through a risk-budget selector:

$$S_{\mathrm{RB}}(y) = \frac{1}{T}\sum_{t=1}^{T}\mu_t - \lambda\frac{1}{T}\sum_{t=1}^{T}\mathbf{1}[\sigma_t > \tau], \qquad \sigma_t = \sqrt{\frac{\mu_t(1-\mu_t)}{\kappa_t + 1}}.$$

It keeps the average reward term, but discounts rollouts that contain many high-uncertainty steps.

BETAPRM achieves the best accuracy in every backbone–benchmark block. Its average gains over standard PRM are $+1.29$, $+1.46$, $+3.37$, and $+2.66$ points for InternVL3-14B, InternVL3-8B, InternVL2.5-8B, and Qwen2.5-VL-7B, respectively. These gains suggest that our Beta-Binomial objective learns useful reward means and concentrations from observed continuation counts, allowing the selector to down-weight high-scoring traces whose rewards are uncertain.

**BETAPRM remains comparable overall in step-level error detection.** Table 2 reports results on VisualProcessBench (Wang et al., 2026), a benchmark for step-level error detection. Each reasoning trace has human step-wise correctness labels, and a PRM score is thresholded into a binary prediction of whether each step is correct or erroneous.

BETAPRM remains competitive with standard PRM under this thresholding setting. Across the evaluated backbones, its overall micro-F1 remains comparable to PRM: it matches PRM on InternVL3-14B and improves slightly on InternVL3-8B and Qwen2.5-VL-7B. Additional analysis shows that the two largest subset regressions are substantially reduced when evaluation stops at the first error, including that step (Appendix A.5). Together with the Best-of-16 results, these findings suggest that BETAPRM improves candidate selection while maintaining comparable overall performance in separating correct and erroneous steps under a standard threshold.

**Training and reliability-aware selection provide complementary gains.** To distinguish the contribution of PRM training from that of inference-time selection, we compare four combinations of training formulations and selectors on the same candidate pools. Both Standard PRM and BETAPRM are evaluated with mean-only selection, which ranks candidates by their average predicted reward. We additionally apply the uncertainty-aware selector to Standard PRM using the proxy $\sigma_t^{\mathrm{proxy}} = \sqrt{\mu_t(1-\mu_t)}$, and to BETAPRM using uncertainty derived from its learned $\kappa_t$.

As shown in Table 3, replacing Standard PRM with BETAPRM under the same mean-only selector improves accuracy by 2.23 points on average across the four settings. Adding the learned-$\kappa$ selector

Table 3: Controlled ablation of PRM training and inference-time selection under Best-of-16. All selectors operate on the same candidate pools. Avg. denotes average final-answer accuracy across the four backbone–benchmark settings.

| PRM training | Selector | InternVL2.5-8B | | Qwen2.5-VL-7B | | Avg. |
|---|---|---|---|---|---|---|
| | | OlympiadBench | MathVision | OlympiadBench | MathVision | |
| Standard PRM | Mean-only | 11.33 | 21.38 | 14.00 | 21.38 | 17.02 |
| Standard PRM | Proxy uncertainty | 12.07 | 22.70 | 15.33 | 22.37 | 18.12 |
| BETAPRM | Mean-only | 14.00 | 24.01 | 15.33 | 23.68 | 19.26 |
| BETAPRM | Learned $\kappa$ | **15.33** | **25.66** | **17.33** | **24.34** | **20.67** |

Table 5: ACA improves the accuracy–token tradeoff in PRM-guided Best-of-16. Token counts are reported in thousands. Percentages indicate token reduction relative to Vanilla BoN.

| Method | Adaptive Allocation | Early Stopping | MathVision | | OlympiadBench | | MathVerse | | MathVista | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Acc. ↑ | Tokens ↓ | Acc. ↑ | Tokens ↓ | Acc. ↑ | Tokens ↓ | Acc. ↑ | Tokens ↓ |
| *InternVL2.5-8B* | | | | | | | | | | |
| Vanilla BoN | × | × | 25.00 | 1383k | 15.33 | 1151k | 44.47 | 17932k | 60.90 | 2790k |
| $\text{ACA}_{\text{w/o EarlyStop}}$ | ✓ | × | 24.01 | 1237k (↓10.56%) | 15.33 | 1028k (↓10.69%) | 42.99 | 14692k (↓18.07%) | 60.20 | 2462k (↓11.76%) |
| ACA | ✓ | ✓ | **26.32** | **965k** (↓30.24%) | **16.67** | **958k** (↓16.76%) | **45.58** | **11912k** (↓33.57%) | **62.20** | **1949k** (↓30.14%) |
| *Qwen2.5-VL-7B* | | | | | | | | | | |
| Vanilla BoN | × | × | 24.67 | 1383k | 16.67 | 1151k | 45.74 | 17932k | 63.30 | 2790k |
| $\text{ACA}_{\text{w/o EarlyStop}}$ | ✓ | × | 24.34 | 1205k (↓12.87%) | 16.67 | 1084k (↓5.82%) | 44.34 | 16075k (↓10.36%) | 62.80 | 2551k (↓8.57%) |
| ACA | ✓ | ✓ | **26.65** | **988k** (↓28.57%) | **18.00** | **928k** (↓19.39%) | **46.40** | **12015k** (↓33.00%) | **64.00** | **2030k** (↓27.22%) |

provides a further 1.41-point gain, giving a total improvement of 3.64 points over Standard PRM with mean-only selection. In contrast, applying the proxy-uncertainty selector to Standard PRM improves accuracy by 1.10 points, and still performs below BETAPRM with mean-only selection. These results show that the gains are not explained by the inference-time selector alone: the BETAPRM training formulation improves candidate ranking under the same selector, while learned reliability provides an additional, complementary benefit. An additional ablation of the auxiliary evidence regularizer is provided in Appendix D.

**Learned concentration captures reliability beyond the predicted mean.** We directly evaluate whether $\kappa$ provides reliability information beyond $\mu$ on held-out prefixes. After matching the predicted $\mu$ distributions of low- and high-$\kappa$ groups, we measure the root mean squared error (RMSE) between $\mu$ and the observed continuation success rate $K/N$. Table 4 shows that high-$\kappa$ predictions have 37.8–51.7% lower RMSE across all four backbones. They also yield narrower predictive intervals at comparable coverage (Appendix C.1). In a separate controlled evaluation that keeps each prefix's $\mu$ unchanged, learned $\kappa$ achieves lower continuation-count negative log-likelihood than both oracle fixed and within-$\mu$ shuffled controls (Appendix C.2). Together, these results provide direct evidence that $\kappa$ captures instance-specific relative reliability beyond the predicted mean.

Table 4: Held-out RMSE at matched $\mu$.

| Backbone | Low $\kappa$ | High $\kappa$ |
|---|---|---|
| InternVL2.5-8B | 0.283 | **0.176** |
| InternVL3-8B | 0.277 | **0.153** |
| InternVL3-14B | 0.242 | **0.117** |
| Qwen2.5-VL-7B | 0.253 | **0.126** |

### 6.3 ACA IMPROVES THE INFERENCE-TIME ACCURACY-TOKEN TRADEOFF

**ACA uses fewer tokens while improving Best-of-$N$ accuracy.** Table 5 compares ACA with fixed-budget PRM-guided Best-of-$N$. All methods use the same maximum budget of $N = 16$ candidate generations and the same BETAPRM risk-budget selector $S_{\text{RB}}$ for final selection; they differ only in how the budget is spent. Vanilla Best-of-$N$ generates all candidates from scratch, whereas ACA spends the budget in stages and uses BETAPRM uncertainty to decide whether to stop or repair uncertain prefixes.

ACA improves the accuracy–token tradeoff across both backbones. Across both InternVL2.5-8B and Qwen2.5-VL-7B, ACA improves all four benchmarks, saving 16.76%–33.57% tokens on

Table 6: ACA ablation under a Best-of-16 budget. BETAPRM improves the accuracy–token tradeoff over Standard PRM with proxy uncertainty or reward-only allocation.

| Method | MathVision | | OlympiadBench | | MathVerse | | MathVista | |
|---|---|---|---|---|---|---|---|---|
| | Acc. | Tokens | Acc. | Tokens | Acc. | Tokens | Acc. | Tokens |
| *InternVL2.5-8B* | | | | | | | | |
| ACA w. BETAPRM (Learned Uncertainty) | 25.99 | 965k | 16.67 | 958k | 45.58 | 11912k | 62.10 | 1949k |
| ACA w. Standard PRM (Proxy Uncertainty) | 22.37 | 1225k | 14.00 | 994k | 44.29 | 14551k | 61.40 | 2304k |
| ACA w. Standard PRM (Reward-Only) | 21.38 | 738k | 14.67 | 527k | 43.02 | 10783k | 58.80 | 1799k |
| *Qwen2.5-VL-7B* | | | | | | | | |
| ACA w. BETAPRM (Learned Uncertainty) | 26.65 | 988k | 18.00 | 928k | 46.40 | 12015k | 64.00 | 2030k |
| ACA w. Standard PRM (Proxy Uncertainty) | 24.67 | 1133k | 15.33 | 915k | 45.41 | 13595k | 62.30 | 2072k |
| ACA w. Standard PRM (Reward-Only) | 21.38 | 604k | 14.67 | 499k | 44.47 | 9138k | 60.30 | 1618k |

InternVL2.5-8B and $19.39\%$–$33.00\%$ on Qwen2.5-VL-7B. The ablation without early stopping shows that adaptive expansion alone mainly reduces computation, but can keep spending budget even after the current selected answer is already reliable, introducing additional candidates that may distract an imperfect PRM selector. The full ACA gives the strongest tradeoff by combining uncertainty-guided expansion with confidence-based stopping. We further report a sensitivity analysis of ACA's control hyperparameters in Appendix B.

**BETAPRM provides a distinct learned uncertainty signal.** We investigate whether the explicit uncertainty modeling in BETAPRM is actually necessary for ACA, or if a standard PRM would suffice to achieve the same efficiency. We compare BETAPRM with ACA variants using a standard PRM. The first baseline, *ACA with Standard PRM (Reward Only)*, removes uncertainty entirely: it ranks candidates by the average process reward from standard PRM, uses a score-margin stopping rule, and repairs the lowest-scoring step when allocating more computation. The second baseline, *ACA with Standard PRM (Proxy Uncertainty)*, uses $\sigma_t = \sqrt{\mu_t(1-\mu_t)}$, as an uncertainty proxy for ACA. Our full variant, *ACA with* BETAPRM *(Learned Uncertainty)*, uses the uncertainty induced by its learned concentration $\kappa_t$. For fair comparison, all variants use the same linear risk-adjusted score $S_{\text{lin}}(y) = \frac{1}{T}\sum_{t=1}^{T}(\mu_t - \lambda\sigma_t)$. For the reward-only baseline, we set $\sigma_t = 0$, so this reduces to the average process reward. This shared form keeps the comparison well-defined across variants and focuses the ablation on the source of uncertainty.

As shown in Table 6, BETAPRM with learned uncertainty gives the best accuracy–token tradeoff. Across the evaluated backbones, it improves over proxy uncertainty in both dimensions, achieving higher accuracy while using fewer tokens in all evaluated settings. In contrast, the $\mu$-only variant often uses the fewest tokens, but at a clear accuracy cost. Without a reliability estimate, it treats low reward as the only reason to repair and large reward margins as sufficient evidence to stop. This misses the key cases ACA is designed for: prefixes whose score is not necessarily low, but whose PRM score is uncertain enough that additional continuations could change the selected answer. Thus, reward-only allocation saves computation by reducing exploration, but loses accuracy because it cannot identify where uncertainty still matters.

## 7 CONCLUSION

We study how PRMs can score reasoning steps while also indicating when those scores should be trusted. We propose BETAPRM, a distributional PRM that represents each reasoning prefix with a Beta belief over its success probability and trains it from Monte Carlo observations using a Beta-Binomial objective. This gives the model both a predicted prefix success probability and a learned reliability estimate for that prediction. Experiments show that BETAPRM improves PRM-guided Best-of-$N$ selection while maintaining comparable overall step-level error-detection performance. Using this reliability signal, Adaptive Computation Allocation further improves final-answer accuracy while reducing inference tokens by up to $33.57\%$. Overall, BETAPRM turns scalar process rewards into reliability-aware signals for test-time selection and computation allocation.

### AI USE STATEMENT

Generative AI was used in the research workflow to generate reasoning trajectories and is also a core component of the proposed method, as described in the paper. AI-generated research outputs were assessed using the benchmark evaluation procedures reported in the paper. Generative AI was additionally used for limited language editing. The authors reviewed the final manuscript and take responsibility for all AI-assisted content, claims, and results.

### REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we describe the experimental setup in Section 6 and provide details of the training data, backbone configurations, optimization settings, and hyperparameters in Appendix A. The Best-of-$N$ evaluation, step-level error detection, and adaptive computation allocation protocols are specified in Appendix A.3, Appendix A.4, and Appendix A.6, respectively.

## REFERENCES

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv e-prints*, pp. arXiv–2502, 2025.

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

Jie Cao and Jing Xiao. An augmented benchmark dataset for geometric question answering through dual parallel text encoding. In *Proceedings of the 29th international conference on computational linguistics*, pp. 1511–1520, 2022.

Hyungjoo Chae, Sunghwan Kim, Junhee Cho, Seungone Kim, Seungjun Moon, Gyeom Hwangbo, Dongha Lim, Minjin Kim, Yeonjun Hwang, Minju Gwak, Dongwook Choi, Minseok Kang, Gwanhoon Im, ByeongUng Cho, Hyojun Kim, Jun Hee Han, Taeyoon Kwon, Minju Kim, Beong woo Kwak, Dongjin Kang, and Jinyoung Yeo. Web-shepherd: Advancing PRMs for reinforcing web agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=G2kMroO9UV.

Shuaichen Chang, David Palzer, Jialin Li, Eric Fosler-Lussier, and Ningchuan Xiao. MapQA: A dataset for question answering on choropleth maps. In *NeurIPS 2022 First Table Representation Workshop*, 2022. URL https://openreview.net/forum?id=znKbVjeR0yI.

Jiaqi Chen, Tong Li, Jinghui Qin, Pan Lu, Liang Lin, Chongyu Chen, and Xiaodan Liang. Unigeo: Unifying geometry logical reasoning via reformulating mathematical expression. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3313–3323, 2022.

Qiguang Chen, Libo Qin, Jin Zhang, Zhi Chen, Xiao Xu, and Wanxiang Che. M3cot: A novel benchmark for multi-domain multi-step multi-modal chain-of-thought. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8199–8221, 2024a.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024b.

Zhengyu Chen, Yudong Wang, Teng Xiao, Ruochen Zhou, Xusheng Yang, Wei Wang, Zhifang Sui, and Jingang Wang. From mathematical reasoning to code: Generalization of process reward models in test-time scaling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 30368–30376, 2026.

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

Ning Dai, Zheng Wu, Renjie Zheng, Ziyun Wei, Wenlei Shi, Xing Jin, Guanlin Liu, Chen Dun, Liang Huang, and Lin Yan. Process supervision-guided policy optimization for code generation. *arXiv preprint arXiv:2410.17621*, 2024.

Lingxiao Du, Fanqing Meng, Zongkai Liu, Zhixiang Zhou, Ping Luo, Qiaosheng Zhang, and Wenqi Shao. Mm-prm: Enhancing multimodal mathematical reasoning with scalable step-level supervision. *arXiv preprint arXiv:2505.13427*, 2025.

Keyu Duan, Zichen Liu, Xin Mao, Tianyu Pang, Changyu Chen, Qiguang Chen, Michael Qizhe Shieh, and Longxu Dou. Efficient process reward model training via active learning. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=CJ2FmPmoDE.

Jiahui Gao, Renjie Pi, Jipeng Zhang, Jiacheng Ye, Wanjun Zhong, Yufei Wang, Lanqing HONG, Jianhua Han, Hang Xu, Zhenguo Li, and Lingpeng Kong. G-LLaVA: Solving geometric problem with multi-modal large language model. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=px1674Wp3C.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6904–6913, 2017.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rstar-math: Small LLMs can master math reasoning with self-evolved deep thinking. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=5zwF1GizFa.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, 2024a.

Mingqian He, Yongliang Shen, Wenqi Zhang, Zeqi Tan, and Weiming Lu. Advancing process verification for large language models via tree-based preference learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 2086–2099, 2024b.

Vlad Hosu, Hanhe Lin, Tamas Sziranyi, and Dietmar Saupe. Koniq-10k: An ecologically valid database for deep learning of blind image quality assessment. *IEEE Transactions on Image Processing*, 29:4041–4056, 2020.

Pengfei Hu, Zhenrong Zhang, Qikai Chang, Shuhang Liu, Jiefeng Ma, Jun Du, Jianshu Zhang, Quan Liu, Jianqing Gao, Feng Ma, et al. Prm-bas: Enhancing multimodal reasoning through prm-guided beam annealing search. *arXiv preprint arXiv:2504.10222*, 2025.

Chengsong Huang, Langlin Huang, Jixuan Leng, Jiacheng Liu, and Jiaxin Huang. Efficient test-time scaling via self-calibration. *arXiv preprint arXiv:2503.00031*, 2025.

Zheng Huang, Kai Chen, Jianhua He, Xiang Bai, Dimosthenis Karatzas, Shijian Lu, and CV Jawahar. Icdar2019 competition on scanned receipt ocr and information extraction. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1516–1520. IEEE, 2019.

Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2901–2910, 2017.

Kushal Kafle, Brian Price, Scott Cohen, and Christopher Kanan. Dvqa: Understanding data visualizations via question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5648–5656, 2018.

Samira Ebrahimi Kahou, Vincent Michalski, Adam Atkinson, Ákos Kádár, Adam Trischler, and Yoshua Bengio. Figureqa: An annotated figure dataset for visual reasoning. *arXiv preprint arXiv:1710.07300*, 2017.

Mehran Kazemi, Hamidreza Alvari, Ankit Anand, Jialin Wu, Xi Chen, and Radu Soricut. Geomverse: A systematic evaluation of large models for geometric reasoning. In *AI for Math Workshop@ ICML 2024*, 2024.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. In *European conference on computer vision*, pp. 235–251. Springer, 2016.

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. Process reward models that think. *arXiv preprint arXiv:2504.16828*, 2025.

Seungone Kim, Ian Wu, Jinu Lee, Xiang Yue, Seongyun Lee, Mingyeong Moon, Kiril Gashteovski, Carolin Lawrence, Julia Hockenmaier, Graham Neubig, et al. Scaling evaluation-time compute with reasoning models as process evaluators. *arXiv preprint arXiv:2503.19877*, 2025.

Jinyuan Li, Chengsong Huang, Langlin Huang, Shaoyang Xu, Haolin Liu, Wenxuan Zhang, and Jiaxin Huang. Training data efficiency in multimodal process reward models. *arXiv preprint arXiv:2602.04145*, 2026.

Zhuowan Li, Xingrui Wang, Elias Stengel-Eskin, Adam Kortylewski, Wufei Ma, Benjamin Van Durme, and Alan L Yuille. Super-clevr: A virtual benchmark to diagnose domain robustness in visual reasoning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14963–14973, 2023.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Haolin Liu, Dian Yu, Sidi Lu, Yujun Zhou, Rui Liu, Zhenwen Liang, Haitao Mi, Chen-Yu Wei, and Dong Yu. Save the good prefix: Precise error penalization via process-supervised rl to enhance llm reasoning. *arXiv preprint arXiv:2601.18984*, 2026.

Runze Liu, Junqi Gao, Jian Zhao, Kaiyan Zhang, Xiu Li, Biqing Qi, Wanli Ouyang, and Bowen Zhou. Can 1b LLM surpass 405b LLM? rethinking compute-optimal test-time scaling. In *Workshop on Reasoning and Planning for Large Language Models*, 2025a. URL https://openreview.net/forum?id=CvjX9Lhpze.

Wei Liu, Junlong Li, Xiwen Zhang, Fan Zhou, Yu Cheng, and Junxian He. Diving into self-evolving training for multimodal reasoning. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=X3ikghfWwD.

Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-Chun Zhu. Inter-GPS: Interpretable geometry problem solving with formal language and symbolic reasoning. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *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)*, pp. 6774–6786, Online, August 2021a. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.528. URL https://aclanthology.org/2021.acl-long.528/.

Pan Lu, Liang Qiu, Jiaqi Chen, Tony Xia, Yizhou Zhao, Wei Zhang, Zhou Yu, Xiaodan Liang, and Song-Chun Zhu. IconQA: A new benchmark for abstract diagram understanding and visual language reasoning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021b. URL https://openreview.net/forum?id=uXa9oBDZ9V1.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521, 2022.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=KUNzEQMWU7.

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

Ruilin Luo, Zhuofan Zheng, Yifan Wang, Xinzhe Ni, Zicheng Lin, Songtao Jiang, Yiyao Yu, Chufan Shi, Lei Wang, Ruihang Chu, et al. Unlocking multimodal mathematical reasoning via process reward model. *arXiv preprint arXiv:2501.04686*, 2025.

Qianli Ma, Haotian Zhou, Tingkai Liu, Jianbo Yuan, Pengfei Liu, Yang You, and Hongxia Yang. Let's reward step by step: Step-level reward model as the navigators for reasoning. *arXiv preprint arXiv:2310.10080*, 2023.

Ahmed Masry, Xuan Long Do, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the association for computational linguistics: ACL 2022*, pp. 2263–2279, 2022.

Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 2200–2209, 2021.

Minesh Mathew, Viraj Bagal, Rubèn Tito, Dimosthenis Karatzas, Ernest Valveny, and C.V. Jawahar. Infographicvqa. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 1697–1706, January 2022.

Tej Deep Pala, Panshul Sharma, Amir Zadeh, Chuan Li, and Soujanya Poria. Error typing for smarter rewards: Improving process reward models with error-aware hierarchical supervision. *arXiv preprint arXiv:2505.19706*, 2025.

Young-Jin Park, Kristjan Greenewald, Kaveh Alim, Hao Wang, and Navid Azizan. Know what you don't know: Uncertainty calibration of process reward models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=hzMkfIrdDT.

Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. Solving geometry problems: Combining text and diagram interpretation. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 1466–1476, 2015.

Wenhao Shi, Zhiqiang Hu, Yi Bin, Junhua Liu, Yang Yang, See Kiong Ng, Lidong Bing, and Roy Ka-Wei Lee. Math-llava: Bootstrapping mathematical reasoning for multimodal large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 4663–4680, 2024.

Shweta Singh, Aayan Yadav, Jitesh Jain, Humphrey Shi, Justin Johnson, and Karan Desai. Benchmarking object detectors with coco: A new path forward. In *European Conference on Computer Vision*, pp. 279–295. Springer, 2024.

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

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. Prmbench: A fine-grained and challenging benchmark for process-level reward models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 25299–25346, 2025.

Alane Suhr, Stephanie Zhou, Ally Zhang, Iris Zhang, Huajun Bai, and Yoav Artzi. A corpus for reasoning about natural language grounded in photographs. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 6418–6428, 2019.

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

Haoqin Tu, Weitao Feng, Hardy Chen, Hui Liu, Xianfeng Tang, and Cihang Xie. Vilbench: A suite for vision-language process reward modeling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 6775–6790, 2025.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Jun Wang, Meng Fang, Ziyu Wan, Muning Wen, Jiachen Zhu, Anjie Liu, Ziqin Gong, Yan Song, Lei Chen, Lionel M Ni, et al. Openr: An open source framework for advanced reasoning with large language models. *arXiv preprint arXiv:2410.09671*, 2024a.

Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Houxing Ren, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. Measuring multimodal mathematical reasoning with math-vision dataset. *Advances in Neural Information Processing Systems*, 37:95095–95169, 2024b.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 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)*, pp. 9426–9439, 2024c.

Shuai Wang, Zhenhua Liu, Jiaheng Wei, Xuanwu Yin, Dong Li, and Emad Barsoum. Athena: Enhancing multimodal reasoning with data-efficient process reward models. *arXiv preprint arXiv:2506.09532*, 2025.

Weiyun Wang, Zhangwei Gao, Lianjie Chen, Zhe Chen, Jinguo Zhu, Xiangyu Zhao, Yangzhou Liu, Yue Cao, Shenglong Ye, Xizhou Zhu, Lewei Lu, Haodong Duan, Yu Qiao, Jifeng Dai, and Wenhai Wang. VisualPRM400k: An effective dataset for training multimodal process reward models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=IHyY6vdYZw.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Wei Xiong, Wenting Zhao, Weizhe Yuan, Olga Golovneva, Tong Zhang, Jason Weston, and Sainbayar Sukhbaatar. Stepwiser: Stepwise generative judges for wiser reasoning. *arXiv preprint arXiv:2508.19229*, 2025.

Runyang You, Yongqi Li, Meng Liu, Wenjie Wang, Liqiang Nie, and Wenjie Li. Parallel test-time scaling for latent reasoning models. *arXiv preprint arXiv:2510.07745*, 2025.

Fei Yu, Anningzhe Gao, and Benyou Wang. Ovm, outcome-supervised value models for planning in mathematical reasoning. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 858–875, 2024.

Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Yu Qiao, et al. Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems? In *European Conference on Computer Vision*, pp. 169–186. Springer, 2024.

Renrui Zhang, Xinyu Wei, Dongzhi Jiang, Ziyu Guo, Yichi Zhang, Chengzhuo Tong, Jiaming Liu, Aojun Zhou, Shanghang Zhang, Peng Gao, and Hongsheng Li. MAVIS: Mathematical visual instruction tuning with an automatic data engine. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=MnJzJ2gvuf.

Wenlin Zhang, Xiangyang Li, Kuicai Dong, Yichao Wang, Pengyue Jia, Xiaopeng Li, Yingyi Zhang, Derong Xu, Zhaocheng Du, Huifeng Guo, Ruiming Tang, and Xiangyu Zhao. Process vs. outcome reward: Which is better for agentic RAG reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=h3LlJ6Bh4S.

Yuxin Zhang, Meihao Fan, Ju Fan, Mingyang Yi, Yuyu Luo, Jian Tan, and Guoliang Li. Reward-sql: Boosting text-to-sql via stepwise reasoning and process-supervised rewards. *arXiv preprint arXiv:2505.04671*, 2025b.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10495–10516, 2025c.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Processbench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1009–1024, 2025.

Tong Zheng, Chengsong Huang, Runpeng Dai, Yun He, Rui Liu, Xin Ni, Huiwen Bao, Kaishen Wang, Hongtu Zhu, Jiaxin Huang, et al. Parallel-probe: Towards efficient parallel thinking via 2d probing. *arXiv preprint arXiv:2602.03845*, 2026.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025.

Table 7: Source coverage of VisualPRM400K-v1.1 used for PRM training.

| Group | Representative sources |
|---|---|
| Diagram / Synthetic Reasoning | AI2D (Kembhavi et al., 2016), CLEVR (Johnson et al., 2017), Super-CLEVR (Li et al., 2023), NLVR2 (Suhr et al., 2019), FigureQA (Kahou et al., 2017), IconQA (Lu et al., 2021b) |
| Chart / Document / OCR QA | ChartQA (Masry et al., 2022), DocVQA (Mathew et al., 2021), DVQA (Kafle et al., 2018), InfographicVQA (Mathew et al., 2022), SROIE (Huang et al., 2019) |
| General VQA and Visual Reasoning | VQAv2 (Goyal et al., 2017), COCO-ReM (Singh et al., 2024), KonIQ-10k (Hosu et al., 2020), M3CoT (Chen et al., 2024a), MAPQA-SUV (Chang et al., 2022) |
| Science and Math Reasoning | ScienceQA (Lu et al., 2022), MathV360K (Shi et al., 2024), MAVIS variants (Zhang et al., 2025a) |
| Geometry Reasoning | Geo170K (Gao et al., 2025), Geometry3K (Lu et al., 2021a), GeoQA+ (Cao & Xiao, 2022), GEOS (Seo et al., 2015), GeomVerse (Kazemi et al., 2024), UniGeo (Chen et al., 2022) |

# A Experimental Setup and Implementation Details

## A.1 Training Data and Backbones

We train all PRMs on VisualPRM400K-v1.1 (Wang et al., 2026). We use this version because it exposes the raw Monte Carlo supervision needed by our objective: for each supervised reasoning prefix, the dataset reports the number $K$ of successful continuations among $N$ sampled continuations. This allows BetaPRM to train on the count pair $(K, N)$, while the standard PRM baseline is trained with cross-entropy using the empirical ratio $K/N$ as a soft label.

After validity filtering, the corpus contains 565,096 rollouts and 3,174,394 annotated steps. Before any model training, we hold out 3,000 problems containing 16,553 reasoning prefixes for evaluation and train all PRMs on the remaining problems. Across the full corpus, the average solution contains 5.62 reasoning steps, with 27.8 words per step. The corpus covers 38 subsets spanning diagram understanding, chart and document QA, general visual question answering, science reasoning, and mathematical/geometry reasoning.

We instantiate both the standard PRM and BetaPRM with four multimodal backbones: InternVL2.5-8B (Chen et al., 2024b), InternVL3-8B (Zhu et al., 2025), InternVL3-14B (Zhu et al., 2025), and Qwen2.5-VL-7B (Bai et al., 2025). For each backbone, we insert a `<prm>` marker after every reasoning step and supervise only the marker positions. The reward mean $\mu_t$ is computed from the `Yes`/`No` reward-token logits. For BetaPRM, we additionally attach a lightweight linear head on the marker hidden state to predict the concentration $\kappa_t$. Unless otherwise specified, we freeze the vision encoder and fine-tune the language model together with the multimodal projection modules. Training takes about 48 hours on 4 A100 GPUs for the 7B/8B backbones and about 48 hours on 8 A100 GPUs for the 14B backbone.

## A.2 Optimization and Model Hyperparameters

We use the same optimization recipe for the standard PRM baseline and BetaPRM whenever they share the same backbone. The standard PRM is trained with the cross-entropy objective over the `Yes`/`No` reward tokens, while BetaPRM replaces this objective with the Beta-Binomial loss and adds the concentration head. The InternVL2.5-8B, InternVL3-8B, and InternVL3-14B experiments use the same hyperparameters; Qwen2.5-VL-7B uses the same optimization settings with its native image preprocessing. Table 8 summarizes the hyperparameters needed to reproduce training.

For InternVL backbones, we use dynamic image resolution with image size $448$, at most $6$ image patches, down-sampling ratio $0.5$, and drop-path rate $0.4$. For Qwen2.5-VL-7B, we use the native Qwen2.5-VL preprocessing with minimum and maximum pixel counts $784$ and $200{,}704$, respectively. All models insert `<prm>` after each reasoning step and compute $\mu_t$ from the `Yes` and `No` logits at these marker positions.

Table 8: Optimization and model hyperparameters. Beta-Binomial-specific rows apply only to BETAPRM.

| Item | Value |
|---|---|
| Optimizer | AdamW |
| Learning Rate | $1 \times 10^{-5}$ |
| Weight Decay | 0.05 |
| LR Schedule | Cosine Decay with Warmup |
| Warmup Ratio | 0.05 |
| Epochs | 1 |
| Global Batch Size | 512 |
| Max Sequence Length | 8192 |
| Trainable Modules | LLM + multimodal projector; vision encoder frozen |
| $\epsilon$ in Beta Parameters | $1 \times 10^{-6}$ |
| $\kappa_{\min}$ | $1 \times 10^{-3}$ |
| Initial $\kappa$ | 4.0 |
| $L_{\text{reg}}$ Coefficient | $5 \times 10^{-2}$ |
| Concentration-head LR Multiplier | 10.0 |

### A.3 BEST-OF-N EVALUATION PROTOCOL

We evaluate PRMs as solution selectors under a Best-of-16 setting. For each problem, InternVL2.5-8B generates a shared pool of 16 candidate solutions, from which each PRM selects one according to its scoring rule. All selectors are evaluated on the same split of each benchmark.

Candidates are generated through vLLM with temperature 0.7, top-$p = 0.9$, top-$k = 30$, and a maximum of 2,048 generated tokens per candidate. We use the following prompt to obtain explicitly segmented reasoning steps and a separately marked final answer:

**Candidate generation**

**1) Reasoning phase:**

- Analyze the question thoroughly and consider plausible solution strategies.
- Proceed step by step and wrap each reasoning step in `<step>...</step>`.
- Keep each step concise, factual, and focused on the question and the image (if provided).
- Do NOT include any text outside `<step>` tags in this phase.

**2) Final answer:**

- After the reasoning steps, add a single blank line.
- Provide the final answer wrapped in exactly one `<answer>...</answer>` tag.
- The final answer must be self-contained and must NOT reference the reasoning text.
- Output nothing outside the required tags and nothing after `</answer>`.

Question: `{question}`

For PRM scoring, each candidate $y = s_{1:T}$ is formatted by inserting a `<prm>` marker after every reasoning step:

$$\texttt{Question: } x \texttt{ Process: } s_1, \texttt{<prm>}, \ldots, s_T, \texttt{<prm>}.$$

At each marker, the model scores the step using the normalized `Yes` probability over the reward tokens `Yes` and `No`. For the Standard PRM baseline, candidates are ranked by the average reward:

$$S_{\text{PRM}}(y) = \frac{1}{T} \sum_{t=1}^{T} \mu_t.$$

For BETAPRM, we additionally extract the concentration $\kappa_t$ and compute the Beta standard deviation:

$$\sigma_t = \sqrt{\frac{\mu_t(1-\mu_t)}{\kappa_t + 1}}.$$

We rank candidates with the risk-budget selector used in the main experiments:

$$S_{\mathrm{RB}}(y) = \frac{1}{T}\sum_{t=1}^{T}\mu_t - \lambda\frac{1}{T}\sum_{t=1}^{T}\mathbf{1}[\sigma_t > \tau].$$

The penalty weight $\lambda$ and uncertainty threshold $\tau$ are selected exclusively on the validation set described in Section 6.1. We use the same predefined grid for all reported BETAPRM runs: $\lambda \in \{0.2, 0.5, 0.7, 1.0, 1.5\}$, with $\tau$ set to the $q$-quantile of validation step-level uncertainties $\sigma_t$, where $q \in \{0.7, 0.8, 0.9\}$. All ACA and risk-budget selector parameters are fixed before downstream benchmark evaluation. No test-benchmark examples, labels, or evaluation results are used for parameter selection or re-tuning.

For final-answer evaluation, we extract the answer from the `<answer>...</answer>` block and use Qwen2.5-32B-Instruct to determine whether it is semantically equivalent to the reference answer. The judge receives the question, reference answer, and extracted candidate answer through the following prompt:

**Answer correctness evaluation**

You are given a question, the correct answer and a model's answer. Please determine if the model's answer matches the correct answer.
Focus only on the mathematical or semantic correctness of the content. Ignore any differences in formatting, such as LaTeX syntax, symbols, styles, or additional wrappers (e.g., `\boxed`, `$...$`, or similar). Compare only the core mathematical or textual meaning of the model's answer and the correct answer.
Only the correctness of the model's answer matters.
Return only "Yes" if the model's answer is correct or "No" if it is incorrect. Only return "Yes" or "No" with no additional text or formatting.
Question:
`{question}`
`----------`
Correct Answer:
`{correct_answer}`
`----------`
Model's Answer:
`{model_answer}`
`----------`

We report final-answer accuracy as the fraction of problems for which the selected candidate's answer is judged correct. The same answer extraction procedure and correctness judge are used for all selectors.

### A.4 VISUALPROCESSBENCH EVALUATION PROTOCOL

We evaluate step-level error detection on VisualProcessBench (Wang et al., 2026). For each instance, we concatenate the question with the provided step-by-step rationale and insert a `<prm>` marker after every step, using the same input format as PRM training. The model produces one score at each marker, which is then converted into a binary prediction of whether the corresponding step is correct or erroneous. Neutral labels are ignored when computing metrics.

For the Standard PRM baseline, the step score is the normalized `Yes` probability $\mu_t$. For BETAPRM, we use the same reward mean together with the learned concentration to compute $\sigma_t = \sqrt{\mu_t(1-\mu_t)/(\kappa_t+1)}$, and evaluate the risk-adjusted step score $s_t = \mu_t - \lambda\sigma_t$, with $\lambda = 0.5$. This uses the reliability signal in the same direction as our downstream selection experiments: uncertain positive-looking steps are scored more conservatively.

Given a threshold $\tau_{\mathrm{cls}}$, steps with $s_t \geq \tau_{\mathrm{cls}}$ are classified as correct and those below the threshold are classified as erroneous. Following the benchmark protocol (Wang et al., 2026), we choose a single global threshold per model by sweeping $\tau_{\mathrm{cls}}$ and maximizing the overall validation F1. We report the overall score and the per-source macro-F1 breakdown on VisualProcessBench.

Table 9: Macro-F1 differences between BETAPRM and Standard PRM, in percentage points, for the two largest subset regressions in Table 2. "Through first error" retains each trajectory up to and including its first annotated error, and retains error-free trajectories in full. The 95% confidence intervals apply to this restricted evaluation.

| Backbone | Subset | All steps | Through first error | 95% CI |
|---|---|---|---|---|
| InternVL2.5-8B | WeMath | $-4.40$ | $\mathbf{-0.05}$ | $[-2.56, +2.24]$ |
| InternVL3-14B | MMMU | $-3.94$ | $\mathbf{-0.69}$ | $[-1.73, +1.14]$ |

### A.5 Error Detection up to the First Incorrect Step

We further characterize the two largest subset regressions in Table 2 by examining whether they persist when evaluation stops at the first error. For each trajectory, we retain all steps up to and including its first annotated error, while retaining error-free trajectories in full. Thresholds and metrics remain fixed. We report the macro-F1 difference between BETAPRM and Standard PRM under this restricted evaluation and compare it with the difference obtained when evaluating all steps. For the restricted evaluation, we estimate 95% confidence intervals from 10,000 paired trajectory-level bootstrap resamples, using the same resampled trajectories for both models.

As shown in Table 9, the two largest subset regressions are substantially reduced when evaluation stops at the first error. The gap decreases from $-4.40$ to $-0.05$ points on WeMath with InternVL2.5-8B, and from $-3.94$ to $-0.69$ points on MMMU with InternVL3-14B. Both confidence intervals include zero. These results suggest that, for these two cases, the observed gaps are primarily associated with classification after the trajectory has already become incorrect, rather than with classification of the steps leading up to and including its first error.

This distinction is relevant to PRM-guided rejection, truncation, and repair, where identifying an early error can already motivate intervention. We therefore interpret Table 2 as an auxiliary compatibility check, since BETAPRM is trained to model prefix success rather than specifically optimize isolated step classification. Across the five benchmark sources, its overall score matches or exceeds Standard PRM on three of the four backbones, with a 0.57-point decrease on InternVL2.5-8B. The overall results and the restricted evaluation support the conclusion that BETAPRM remains comparable overall in step-level error detection, while its main benefits concern reliability-aware candidate selection and adaptive computation.

### A.6 Adaptive Computation Allocation Details

ACA is evaluated under the same maximum Best-of-16 budget as the fixed-budget baseline. For each problem, ACA first samples $n_0 = 4$ complete candidates from scratch. If the stopping criterion is not satisfied, it allocates another batch of $m = 4$ candidates, up to the maximum budget $N = 16$. All new candidates are generated by InternVL2.5-8B with the same decoding parameters as the fixed-budget Best-of-16 baseline: temperature 0.7, top-$p = 0.9$, top-$k = 30$ and maximum new tokens 2048.

At each stage, candidates are scored by the linear risk-adjusted score used in the ACA stopping rule,

$$S_{\mathrm{lin}}(y) = \frac{1}{T}\sum_{t=1}^{T}(\mu_t - \lambda\sigma_t),$$

with $\lambda = 0.5$. The lower and upper confidence bounds use $U(y) = T^{-1}\sum_t \sigma_t$ with $c_{\mathrm{stop}} = 0.3$. When ACA continues, it expands the highest-UCB non-winner competitor. For prefix repair, we use $p_{\mathrm{bad}} = 0.3$ as the low-quality threshold and $c_{\mathrm{cut}} = 1.0$ in the conservative step score $\mu_t - c_{\mathrm{cut}}\sigma_t$. The uncertainty penalty in $S_{\mathrm{lin}}$, $c_{\mathrm{stop}}$, $p_{\mathrm{bad}}$, and $c_{\mathrm{cut}}$ are selected exclusively on the same validation set used for risk-budget selection and remain fixed during downstream evaluation.

For the main ACA results, final candidate selection uses the same risk-budget selector $S_{\mathrm{RB}}$ as the Best-of-16 evaluation. For the ACA ablation in Table 6, all variants instead use the shared linear score $S_{\mathrm{lin}}$, with $\sigma_t = 0$ for the reward-only Standard PRM baseline. This keeps the ablation focused on the source of uncertainty.

Table 10: Sensitivity of ACA to its control hyperparameters on MathVision with the InternVL2.5-8B reward model. Each row varies one parameter while holding the others fixed. Bold values indicate the default configuration. K denotes thousands of generated tokens.

| Parameter | Values | Accuracy (%) | Tokens |
|---|---|---|---|
| $\lambda$ | 0.3 / **0.5** / 0.7 | 26.32 / **26.32** / 25.99 | 1029K / **965K** / 941K |
| $c_{\text{stop}}$ | 0.2 / **0.3** / 0.4 | 25.65 / **26.32** / 25.99 | 878K / **965K** / 1129K |
| $c_{\text{cut}}$ | 0.8 / **1.0** / 1.2 | 26.32 / **26.32** / 26.32 | 936K / **965K** / 965K |
| $p_{\text{bad}}$ | 0.2 / **0.3** / 0.4 | 25.99 / **26.32** / 26.32 | 930K / **965K** / 965K |

# B Sensitivity to ACA Control Hyperparameters

We conduct a sensitivity study on the MathVision setting in the InternVL2.5-8B block of Table 5, varying one control hyperparameter at a time around the default configuration under the same experimental setup and maximum candidate budget of $N = 16$. We vary the uncertainty penalty $\lambda$ used in ACA's linear candidate score, the stopping coefficient $c_{\text{stop}}$, the cutpoint coefficient $c_{\text{cut}}$, and the low-quality threshold $p_{\text{bad}}$, while keeping the remaining parameters at their default values.

As shown in Table 10, ACA maintains comparable or better accuracy than Vanilla Best-of-16 across all tested configurations while reducing token usage by 18.4%–36.5%. The parameters affect computation through different parts of the allocation procedure. The stopping coefficient $c_{\text{stop}}$ directly controls early stopping and has the largest impact on token usage within the tested ranges. The uncertainty penalty $\lambda$ affects uncertainty-aware candidate ranking and leads to milder changes, while $c_{\text{cut}}$ and $p_{\text{bad}}$ determine discrete repair cutpoints, so their behavior can remain unchanged until a threshold crossing alters the selected cutpoint. Overall, ACA remains effective across the tested range of control settings rather than relying on a narrow operating window.

# C Direct Validation of Learned Reliability

## C.1 High-Concentration Predictions Are More Reliable at Matched Means

We evaluate whether the learned concentration $\kappa$ provides reliability information beyond the predicted mean $\mu$. At matched $\mu$, we find that high-$\kappa$ predictions have lower root mean squared error (RMSE) and sharper predictive intervals at comparable coverage.

**Held-out evaluation setup.** We evaluate 16,553 prefix-level predictions from 3,000 problems that were split from the original training corpus and reserved as a held-out set before any model training. For each prefix, $N$ Monte Carlo continuations are independently sampled to obtain the observed success count $K$. To isolate the contribution of $\kappa$ from that of $\mu$, we define low- and high-$\kappa$ groups as the bottom and top quartiles. We then divide predictions into $\mu$-bins of width $0.002$ and randomly subsample equal numbers of low- and high-$\kappa$ examples within each bin. The resulting groups therefore have nearly identical $\mu$ distributions.

**Evaluation metrics.** We evaluate two aspects of prediction reliability. First, we measure the root mean squared error (RMSE) between the predicted mean $\mu_i$ and the observed continuation success rate $K_i/N$. For each matched group $\mathcal{G}$, either low-$\kappa$ or high-$\kappa$, we compute

$$\text{RMSE}(\mathcal{G}) = \sqrt{\frac{1}{|\mathcal{G}|}\sum_{i\in\mathcal{G}}\left(\mu_i - \frac{K_i}{N}\right)^2}, \tag{1}$$

where $|\mathcal{G}|$ is the number of prefixes in the matched group, $\mu_i$ is the predicted success probability for prefix $i$, and $K_i$ is the number of successful continuations among $N$ independently sampled continuations. Lower RMSE indicates that the predicted mean is closer to the observed continuation success rate.

Second, we evaluate the coverage and average width of the equal-tail $90\%$ Beta-Binomial predictive interval induced by $\alpha_i = \mu_i\kappa_i$ and $\beta_i = (1-\mu_i)\kappa_i$. Coverage measures how often the observed count

Table 11: Reliability evaluation at matched predicted means on held-out prefixes. Low- and high-$\kappa$ groups are matched within $\mu$-bins of width $0.002$. Across all four backbones, high-$\kappa$ predictions have lower RMSE and narrower $90\%$ Beta-Binomial predictive intervals at comparable empirical coverage. Interval widths are measured in successes out of 16.

| **Backbone** | **RMSE ↓** | | **90% Coverage** | | **Interval Width ↓** | |
|---|---|---|---|---|---|---|
| | Low $\kappa$ | High $\kappa$ | Low $\kappa$ | High $\kappa$ | Low $\kappa$ | High $\kappa$ |
| InternVL2.5-8B | 0.283 | **0.176** | 96.1% | 96.9% | 11.69 | **9.03** |
| InternVL3-8B | 0.277 | **0.153** | 98.9% | 98.6% | 11.80 | **8.19** |
| InternVL3-14B | 0.242 | **0.117** | 97.4% | 95.5% | 7.18 | **4.85** |
| Qwen2.5-VL-7B | 0.253 | **0.126** | 97.2% | 98.8% | 9.09 | **7.16** |

$K_i$ falls inside the interval, while interval width measures how wide a range of continuation outcomes the model considers plausible. We report interval widths in units of successful continuations out of 16. Lower RMSE and narrower intervals at comparable coverage indicate more reliable predictions.

**Results.** Table 11 shows that high-$\kappa$ predictions are both more accurate and more precise. Concretely, for InternVL3-14B, the low-$\kappa$ group has an average $90\%$ Beta-Binomial predictive interval width of 7.18 successes out of 16, with the observed $K_i$ falling within these intervals for $97.4\%$ of prefixes. The high-$\kappa$ group narrows the interval to 4.85 while retaining comparable empirical coverage at $95.5\%$. The same pattern holds consistently across all four backbones. High-$\kappa$ predictions have 37.8–51.7% lower RMSE, meaning that their predicted mean $\mu$ is substantially closer to the observed continuation success rate $K/N$, and 21.3–32.4% narrower predictive intervals at comparable coverage.

Coverage alone is not sufficient, since wider intervals can trivially cover more observations. The key result is that high-$\kappa$ predictions are both more accurate and more precise. Because the two groups are matched on $\mu$, these results provide direct out-of-sample evidence that $\kappa$ captures instance-specific relative reliability beyond the predicted mean.

### C.2 Learned Concentration Improves Held-Out Count Likelihood

We further evaluate whether the learned concentration provides an appropriate predictive dispersion around the predicted mean. Using the same held-out reliability set, we keep each prefix's predicted mean $\mu_i$ unchanged across all settings and vary only its concentration $\kappa_i$. Using InternVL2.5-8B, we evaluate the Beta-Binomial negative log-likelihood (NLL) of the observed success counts:

$$\mathcal{L}_{\mathrm{NLL}}(\boldsymbol{\kappa}) = -\frac{1}{M}\sum_{i=1}^{M}\log\left[\binom{N}{K_i}\frac{B(K_i+\mu_i\kappa_i,\, N-K_i+(1-\mu_i)\kappa_i)}{B(\mu_i\kappa_i,\,(1-\mu_i)\kappa_i)}\right], \tag{2}$$

where $M$ is the number of held-out prefixes, $\boldsymbol{\kappa} = (\kappa_1, \ldots, \kappa_M)$ contains their assigned concentrations, $K_i$ is the observed number of successful continuations among $N$ trials, and $B(\cdot,\cdot)$ is the Beta function. Lower NLL indicates better average log predictive probability for the observed continuation counts.

We compare the learned $\kappa_i$ with two controls. The first assigns a single global concentration $\kappa^\star$ to all prefixes, with $\kappa^\star$ chosen to minimize NLL on the held-out split. Since this value is optimized on the same split used for evaluation, we refer to it as the oracle fixed-concentration control. The second partitions prefixes into predicted-$\mu$ bins of width $0.05$ and randomly permutes $\kappa_i$ within each bin. This exactly preserves the empirical distribution of $\kappa$ within each local $\mu$ range while removing its prefix-specific assignment. We repeat the permutation 100 times and report the mean NLL. Both controls leave every prefix's predicted mean unchanged: the fixed control tests whether a single shared concentration suffices, while the shuffled control tests whether assigning concentrations to the appropriate prefixes matters beyond preserving their local distribution.

As shown in Table 12, the learned $\kappa_i$ achieves the lowest held-out NLL. Since each prefix's $\mu_i$ is unchanged and only $\kappa_i$ varies, this directly tests whether the predicted concentration provides an appropriate predictive dispersion around $\mu_i$. The oracle fixed-concentration control performs

Table 12: Held-out continuation-count NLL on InternVL2.5-8B. Each prefix's predicted mean $\mu_i$ is identical across all settings; only the assigned concentration changes. The fixed concentration is optimized on the evaluation split, and the shuffled result is averaged over 100 permutations.

| Concentration assignment | Beta-Binomial NLL ↓ |
|---|---|
| Learned $\kappa_i$ | **1.2629** |
| Oracle fixed $\kappa^\star$ | 1.3664 |
| Shuffled $\kappa_i$ within $\mu$ bins | 1.3726 |

Table 13: Ablation of the auxiliary evidence regularizer on InternVL2.5-8B under PRM-guided Best-of-16 selection. Removing $L_{\text{reg}}$ consistently reduces accuracy.

| Method | MathVision | OlympiadBench | MathVerse | MathVista | Avg. |
|---|---|---|---|---|---|
| BETAPRM | 25.66 | 15.33 | 44.31 | 61.30 | 36.65 |
| BETAPRM w/o $L_{\text{reg}}$ | 24.67 (↓ 0.99) | 14.00 (↓ 1.33) | 43.63 (↓ 0.68) | 60.20 (↓ 1.10) | 35.63 (↓ 1.02) |

worse, showing that a single shared concentration does not explain the observed counts as well as the learned prefix-specific concentrations. The within-$\mu$ shuffle also worsens NLL, even though the predicted means and the local distribution of $\kappa$ are unchanged. Together, these results show that the prefix-specific assignment of concentration contributes to predictive quality, providing direct evidence that $\kappa$ captures reliability information beyond the predicted mean.

## D Ablation of the Auxiliary Evidence Regularizer

We evaluate the contribution of the auxiliary evidence regularizer $L_{\text{reg}}$ by comparing the full BETAPRM training objective with a variant trained using only the Beta-Binomial likelihood. Both variants use the InternVL2.5-8B backbone and are evaluated under PRM-guided Best-of-16 selection on MathVision, OlympiadBench, MathVerse, and MathVista.

Table 13 shows that adding $L_{\text{reg}}$ to the Beta-Binomial likelihood improves all four Best-of-16 benchmarks, with an average gain of 1.02 points. This is consistent with its intended role: when the predicted mean $\mu_t$ disagrees with the observed Monte Carlo ratio $K_t/N$, the regularizer penalizes high concentration. The stop-gradient operation prevents this auxiliary term from directly pulling $\mu_t$ toward $K_t/N$ through the disagreement term, and instead directs its gradient through the concentration prediction $\kappa_t$. The consistent gains support the regularizer's contribution to reliability-aware candidate selection.

## E Training Dynamics of Learned Concentration

We inspect how the learned concentration $\kappa_t$ evolves during training by tracking its mean and 90th percentile across the four backbones. Figure 4 shows that both statistics decrease sharply early in training and subsequently recover, with backbone-dependent fluctuations. This indicates that the predicted Beta beliefs initially become less concentrated and later regain concentration. The recovery is not monotonic: InternVL3-8B and InternVL2.5-8B also exhibit pronounced intermediate drops.

After the initial decrease, the 90th percentile generally remains above the mean, indicating an upper tail of prefixes with higher predicted concentration. These aggregate statistics describe the evolution of concentration during optimization, but do not establish whether higher concentration corresponds to more accurate predictions. We directly evaluate that relationship through the matched-$\mu$ analysis in Appendix C.1 and the controlled count-likelihood evaluation in Appendix C.2.

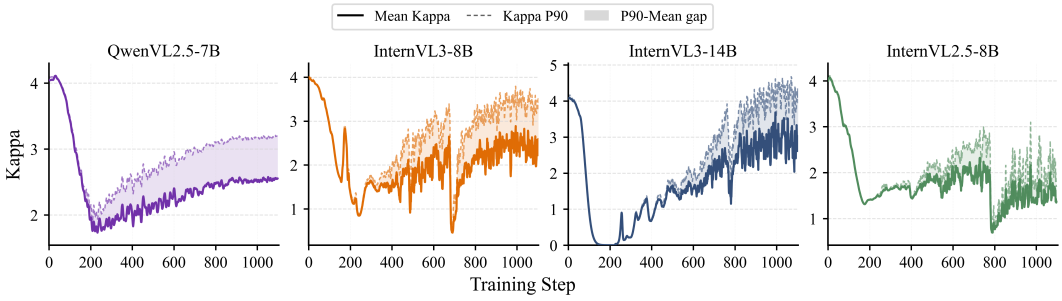


Figure 4: Training dynamics of the learned concentration $\kappa_t$ across four backbones. Solid and dashed lines show the mean and 90th percentile, respectively, and shaded regions indicate their difference. Both statistics decrease early in training and subsequently recover, with different trajectories across backbones.