PROCESS REWARD MODELING WITH RELIABILITY- AND CONSERVATISM-AWARE BAYESIAN BELIEF

Anonymous authors
Paper under double-blind review

ABSTRACT

Process Reward Models (PRMs) play a central role in test-time scaling for large language model (LLM) reasoning. However, PRMs may provide inaccurate process rewards for test-time scaling, which we identify as arising from two sources of uncertainty: (1) *PRM uncertainty* that arises from insufficient supervision data and imperfect PRM training; and (2) *policy uncertainty* that results from the mismatch between the policy to collect supervision data and the policy for test-time scaling, leading to systematic reward overestimation. Previous studies fail to jointly consider both uncertainties. To bridge this gap, we propose BayesianPRM, which treats the process reward as a random variable within a reward hypothesis space and maintains a Bayesian belief over it. Since exact Bayesian inference over the continuous hypothesis space is generally intractable, we discretize the reward hypothesis space with a reward ensemble and introduce two complementary belief updates, each targeting one source of uncertainty: (1) to address PRM uncertainty, we introduce *reliability-aware belief updating* that measures how well each ensemble member explains the supervision data and updates the belief with an amortized belief network optimized by the evidence lower bound (ELBO); and (2) to tackle policy uncertainty, we propose *conservatism-aware belief calibration* which further reweights the belief toward lower rewards to suppress overestimation. These two updates yield a reliability- and conservatism-aware Bayesian belief, whose expectation determines the process reward. Experiments on multimodal reasoning benchmarks indicate that BayesianPRM consistently yields more accurate process reward estimates while enhancing test-time scaling performance.

1 INTRODUCTION

Test-time scaling effectively unlocks the reasoning potential of large language models (LLMs) (Snell et al., 2024; Bi et al., 2024), enabling smaller models to match or even surpass larger ones on challenging mathematics and coding tasks (Liu et al., 2025; Kang et al., 2026). Among common approaches such as Best-of-N (Brown et al., 2024), Process Reward Models (PRMs) (Lightman et al., 2024; Shao et al., 2024; Li et al., 2026b) play a central role by providing step-level estimates of the probability of eventually reaching a correct answer. However, PRMs can produce inaccurate rewards, especially on complex and out-of-distribution (OOD) tasks (Park et al., 2026; Li et al., 2026b), suggesting that test-time scaling should explicitly account for *uncertainty* in process rewards. However, previous studies have not comprehensively investigated such uncertainty.

To bridge this gap, we first analyze the uncertainty in process rewards for test-time scaling and categorize it into two distinct sources. The first is *PRM uncertainty*, which arises from potentially insufficient supervision data and imperfect training procedures, limiting the generalization of PRMs to unseen problems; the second is *policy uncertainty*, which arises from the mismatch between the policies used for PRM training and test-time scaling. For example, the Monte-Carlo labels used for PRM training are often generated by a strong policy, causing the PRM to learn the success probabilities associated with that strong policy. However, for test-time scaling, the policy is often weaker. Consequently, the process reward could be systematically overestimated. Park et al. (2026) and Ma et al. (2026) first observe the reward overestimation phenomenon, and employ quantile regression (Koenker & Hallock, 2001) and conditional optimal transport (Bunne et al., 2022) on a calibration set to address it. We refer to the two approaches as QR-PRM and OT-PRM, respectively. They primarily address policy uncertainty, but do not explicitly account for PRM uncertainty. Be-

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.

Table 1: Comparison between BayesianPRM and relevant recent works.

	PRM Uncertainty	Policy Uncertainty	Technique
BetaPRM (Li et al., 2026b)	✓	✗	Beta Distribution
QR-PRM (Park et al., 2026)	✗	✓	Quantile Regression
OT-PRM (Ma et al., 2026)	✗	✓	Optimal Transport
BayesianPRM (Ours)	✓	✓	Bayesian Learning

taPRM (Li et al., 2026b) models the process reward with a Beta distribution, thereby capturing PRM uncertainty, while leaving policy uncertainty unaddressed. These raise a natural question: *can we incorporate both PRM and policy uncertainty into process reward modeling for test-time scaling?*

Inspired by Bayesian learning (Neal, 2012), We propose BayesianPRM to jointly manage PRM and policy uncertainty. BayesianPRM treats the process reward as a random variable and maintains a belief distribution over the plausible reward hypothesis space. Since exact Bayesian inference over the continuous reward space is generally intractable, we discretize this space with a reward ensemble. We then introduce two stages of belief updates, each tackling one of the two uncertainties. First, to address PRM uncertainty, we introduce *reliability-aware belief updating*. This stage measures how faithfully each ensemble member explains the supervision data. The update is performed with an amortized belief network trained to maximize an evidence lower bound (ELBO). Second, to tackle policy uncertainty, we propose *conservatism-aware belief calibration*. By reweighting the belief toward lower reward values, this stage explicitly suppresses reward overestimation. These two updates yield a reliability- and conservatism-aware Bayesian belief, whose expectation defines the final process reward. We compare BayesianPRM and relevant recent works in Table 1. Our contributions are summarized as follows.

- We identify the uncertainty of PRMs in test-time scaling and categorize it into two sources: PRM uncertainty and policy uncertainty.
- We propose BayesianPRM to address PRM and policy uncertainty through a two-stage belief updating framework: reliability-aware belief updating and conservatism-aware belief calibration.
- Experiments across multimodal reasoning benchmarks indicate that BayesianPRM consistently yields more accurate process reward estimates while enhancing test-time scaling performance.

2 Preliminaries

Process Reward Models. For an input problem x, let $s_{1:T} = (s_1, \ldots, s_T)$ denote a step-by-step solution, where s_t is the t-th reasoning step. Given a prefix $c_t = (x, s_{\le t})$ and policy π, define $q_t^\pi = \Pr(\text{final answer is correct} \mid c_t, \pi)$ as the probability that π successfully completes c_t. A PRM parameterized by θ predicts $r_\theta(c_t) \in [0,1]$ to approximate q_t^π. Since q_t^π is unobservable, PRMs typically learn from its Monte-Carlo estimate $\hat{q}_t^\pi$. For each prefix c_t, we sample N_t continuations from π, among which K_t are successful, yielding $\hat{q}_t^\pi = K_t/N_t$. Letting $\mathcal{D}_{\text{train}} = \{(c_t, \hat{q}_t^\pi)\}_{t=1}^T$ denote the training dataset, PRMs typically optimize the following cross-entropy loss:

$$\mathcal{L}_{\text{CE}}(\theta) = -\frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(c_t, \hat{q}_t^\pi) \in \mathcal{D}_{\text{train}}} \left[\hat{q}_t^\pi \log r_\theta(c_t) + (1 - \hat{q}_t^\pi) \log(1 - r_\theta(c_t))\right]. \tag{1}$$

Bayesian Learning. Bayesian learning (Neal, 2012) provides a principled framework for uncertainty modeling. Given an input-label pair (x, y) and a predictive model $p_\theta(y|x)$, it maintains a belief distribution over hypotheses $\theta \in \Theta$. Starting from a prior $p(\theta)$, the posterior after observing data $\mathcal{D}$ is obtained by Bayes' rule:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{\int_\Theta p(\mathcal{D}|\theta')p(\theta')d\theta'} \propto p(\mathcal{D}|\theta)p(\theta). \tag{2}$$

The posterior represents uncertainty over plausible hypotheses, which could be incorporated into the final prediction by marginalization:

$$p(y|x, \mathcal{D}) = \int_\Theta p_\theta(y|x)p(\theta|\mathcal{D})d\theta. \tag{3}$$

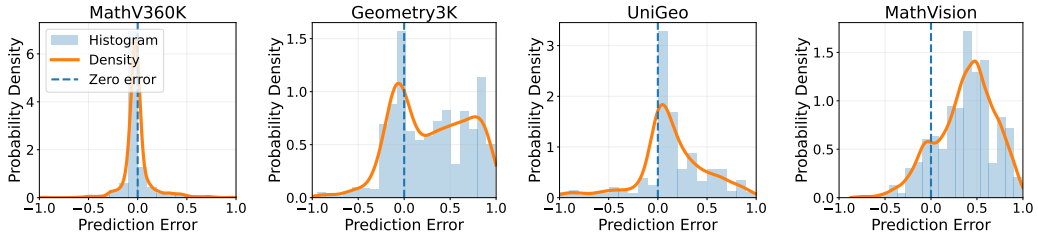


Figure 1: **Illustration of PRM uncertainty and policy uncertainty. (a)-(b)** Prediction error distributions on MathV360K (ID) and Geometry3K (OOD), reflecting PRM uncertainty. **(c)-(d)** Prediction error distribution on UniGeo (ID) and MathVision (OOD), illustrating policy uncertainty. Prediction error is defined as the predicted reward minus the empirical success probability.

3 Analysis of Two Sources of Uncertainty

We first distinguish two sources of uncertainty in process reward estimation: *PRM uncertainty* and *policy uncertainty*. Let π_1 denote the Monte-Carlo rollout policy used to construct the PRM supervision data, and π_2 the policy used for test-time scaling. For a reasoning prefix c_t, the PRM prediction error under π_2 can be decomposed as

$$r_\theta(c_t) - q_t^{\pi_2} = \underbrace{r_\theta(c_t) - q_t^{\pi_1}}_{\text{PRM uncertainty}} + \underbrace{q_t^{\pi_1} - q_t^{\pi_2}}_{\text{policy uncertainty}}, \qquad \forall\, c_t, \tag{4}$$

where $q_t^{\pi_1}$ and $q_t^{\pi_2}$ are the success probability defined in Section 2. The first term characterizes the PRM prediction error under the same policy used to construct its supervision data. Such errors may arise from insufficient supervision and imperfect training, limiting the PRM's generalization to unseen problems. We refer to this source as *PRM uncertainty*, which is related to epistemic uncertainty. The second term captures the change in the true continuation success probability caused by the mismatch between π_1 and π_2, which we refer to as *policy uncertainty*.

We next empirically validate the two sources of uncertainty under both in-distribution (ID) and out-of-distribution (OOD) settings. To examine PRM uncertainty, we use InternVL3-8B (Zhu et al., 2025) as the base model and attach an MLP to its final layer as the PRM to predict a scalar process reward. We train the PRM on VisualPRM400K-v1.1 (Wang et al., 2026), excluding the Geometry3K (Lu et al., 2021) subset for subsequent OOD evaluation. For each remaining subset, 80% of the data is used as the training dataset $\mathcal{D}_{\text{train}}$ and the remaining 20% as the validation dataset $\mathcal{D}_{\text{val}}$. For the ID setting, we evaluate on $\mathcal{D}_{\text{val}}$ of MathV360K (Shi et al., 2024), which can be regarded as independent and identical (i.i.d.) with $\mathcal{D}_{\text{train}}$. For the OOD setting, we evaluate on the held-out Geometry3K subset. Therefore, policy uncertainty is removed in both settings. As shown in Figure 1 (a) and (b), prediction errors on MathV360K are centered around zero but still exhibit variance, whereas the prediction errors on Geometry3K are substantially larger. This indicates that the PRM may remain unreliable on unseen problems, with the issue becoming more pronounced in OOD scenarios, reflecting PRM uncertainty.

We next examine policy uncertainty by using a weak rollout policy to estimate the continuation success probability. We use InternVL2.5-2B (Chen et al., 2024b) as the rollout policy and evaluate both an ID setting, $\mathcal{D}_{\text{val}}$ of UniGeo (Chen et al., 2022), and an OOD setting, MathVision (Wang et al., 2024a). For MathVision, we extract reasoning prefixes from responses generated by the rollout policy. For each prefix in both settings, we generate 16 continuations with the rollout policy and judge their correctness against the ground-truth answer using Qwen2.5-32B-Instruct (Team, 2024). The resulting success rate is used as the empirical success probability. As shown in Figure 1 (c) and (d), the prediction error exhibits a positive bias under both ID and OOD settings, consistent with the observations reported in prior work (Park et al., 2026), providing empirical evidence of policy uncertainty. The bias is more pronounced on MathVision, indicating that reward overestimation is more significant under the OOD setting.

4 METHODOLOGY

In this section, we formally present our BayesianPRM method. In Section 4.1, we formulate process reward prediction as Bayesian learning. In Section 4.2, we give a detailed demonstration of BayesianPRM. In Section 4.3, we extend the discussion to the continuous hypothesis space. In Section 4.4, we present the practical implementation of BayesianPRM.

4.1 PROCESS REWARD MODELING AS BAYESIAN LEARNING

In this section, we formulate process reward modeling from a Bayesian learning perspective. Specifically, we treat the reward hypothesis as a random variable over a hypothesis space Θ. Given a reasoning prefix c_t, each hypothesis $\theta \in \Theta$ predicts a process reward $r_\theta(c_t) = \mathbb{P}(Y_t = 1|c_t, \theta)$, where $Y_t \in \{0, 1\}$ indicates whether a continuation from c_t eventually reaches a correct answer. Let $\mathcal{D}_{\text{train}} = \{(c_t, K_t, N_t)\}_{t=1}^{T}$ denote the PRM training dataset. Let $p(\theta)$ be the prior belief over the hypothesis space Θ. Given the observed supervision and the likelihood $p(\mathcal{D}_{\text{train}}|\theta, c_t)$, the posterior belief $p(\theta|\mathcal{D}_{\text{train}}, c_t)$ is obtained via Bayes' rule as in Equation 2. Given a reasoning prefix c_t, the posterior process reward prediction is obtained by marginalizing over plausible reward hypotheses: $\tilde{r}(c_t) = \int_\Theta r_\theta(c_t)p(\theta|\mathcal{D}_{\text{train}}, c_t)d\theta$. Obtaining $\tilde{r}(c_t)$ requires posterior inference over the continuous hypothesis space Θ, which is generally intractable. Therefore, we consider the discrete hypothesis space with a finite support set of hypotheses $\{\theta_m\}_{m=1}^{M}$. Specifically,

$$p(\theta|\mathcal{D}_{\text{train}}, c_t) = \sum_{m=1}^{M} \alpha_m(c_t)\delta(\theta - \theta_m), \quad \sum_{m=1}^{M} \alpha_m(c_t) = 1, \quad \alpha_m(c_t) \geq 0. \tag{5}$$

Substituting Equation 5 into $\tilde{r}(c_t)$, we have $\tilde{r}(c_t) = \boldsymbol{\alpha}(c_t)^\top \boldsymbol{r}(c_t)$, where $\boldsymbol{\alpha}(c_t) = [\alpha_1(c_t), \ldots, \alpha_M(c_t)]^\top$ and $\boldsymbol{r}(c_t) = [r_1(c_t), \ldots, r_M(c_t)]^\top$. Equation 5 replaces the intractable integration with a tractable weighted sum and naturally admits a reward ensemble implementation (Coste et al., 2024; Duan et al., 2025), where each ensemble member $r_m(\cdot)$ corresponds to a distinct reward hypothesis $\theta_m \in \Theta$.

4.2 RELIABILITY- AND CONSERVATISM-AWARE BAYESIAN BELIEF

In this section, we present BayesianPRM, which incorporates PRM and policy uncertainty into process reward prediction by maintaining a prefix-conditioned Bayesian belief over ensemble reward hypotheses. BayesianPRM performs two successive belief updates: *reliability-aware belief updating* accounts for PRM uncertainty by estimating hypothesis reliability from supervision data, while *conservatism-aware belief calibration* addresses policy uncertainty by mitigating reward overestimation through conservative reweighting.

Reliability-Aware Belief Updating. Recall that in Section 4.1, the process reward is formulated as a weighted sum of reward hypotheses, $\tilde{r}(c_t) = \boldsymbol{\alpha}(c_t)^\top \boldsymbol{r}(c_t)$. The key challenge is to determine the belief weights $\boldsymbol{\alpha}(c_t)$. A simple choice is a uniform prior, $\boldsymbol{\alpha}^{\text{prior}}(c_t) = [\frac{1}{M}, \ldots, \frac{1}{M}]^\top$. However, this ignores PRM uncertainty, since different reward hypotheses may have different reliability for the same prefix. We therefore introduce *reliability-aware belief updating*, which assigns greater belief mass to more reliable reward hypotheses. For prefixes in the training dataset with Monte-Carlo labels, the posterior belief can be obtained from Bayes' rule in the finite hypothesis space. However, such Monte-Carlo labels are unavailable for unseen prefixes at test time. We therefore adopt amortized variational inference and train a belief network w_ν to predict the prefix-conditioned posterior belief. Specifically, for a prefix c_t, the belief network w_ν outputs a reliability-aware belief vector $\boldsymbol{\alpha}^{\text{rel}}(c_t) = \left[\alpha_1^{\text{rel}}(c_t), \ldots, \alpha_M^{\text{rel}}(c_t)\right]^\top$, where $\sum_{i=m}^{M} \alpha_m^{\text{rel}}(c_t) = 1$ and $\alpha_m^{\text{rel}}(c_t) \geq 0$. Intuitively, we expect w_ν to predict the reliability of each reward ensemble member for the given prefix. Taking $\boldsymbol{\alpha}^{\text{prior}}(c_t)$ as the prior belief, we further define the reliability likelihood function to perform amortized variational inference. Specifically, we use a validation set $\mathcal{D}_{\text{val}} = \{(c_t, K_t, N_t)\}_{t=1}^{T'}$, which is independent of the PRM training set $\mathcal{D}_{\text{train}}$. Since $r_m(c_t)$ represents the predicted success probability under θ_m, K_t follows a binomial distribution: $(K_t \mid N_t, c_t, \theta_m) \sim \text{Binomial}(N_t, r_m(c_t))$. Therefore, the reliability likelihood of the supervision data in $\mathcal{D}_{\text{val}}$ under θ_m is: $p(K_t \mid N_t, c_t, \theta_m) = \binom{N_t}{K_t} r_m(c_t)^{K_t} \left(1 - r_m(c_t)\right)^{N_t - K_t}$. This likelihood measures how well $r_m(c_t)$ explains the supervision data, and thus serves as a measure of its reliability for prefix c_t.

Denote $\ell_{t,m} \triangleq \log p(K_t|N_t, c_t, \theta_m) = \log \binom{N_t}{K_t} + K_t \log r_m(c_t) + (N_t - K_t) \log (1 - r_m(c_t))$ as the log-likelihood, we perform amortized variational inference, training the belief network on $\mathcal{D}_{\text{val}}$ by minimizing the following negative evidence lower bound (ELBO):

$$\begin{aligned}\mathcal{L}_{\text{rel}}(\nu) &= -\frac{1}{T'} \sum_{t=1}^{T'} \left[\sum_{m=1}^{M} \alpha_m^{\text{rel}}(c_t) \ell_{t,m} - \beta_1 \cdot D_{\text{KL}}(\boldsymbol{\alpha}^{\text{rel}}(c_t) \| \boldsymbol{\alpha}^{\text{prior}}(c_t)) \right] \\ &\doteq -\frac{1}{T'} \sum_{t=1}^{T'} \sum_{m=1}^{M} \alpha_m^{\text{rel}}(c_t) \left(K_t \log r_m(c_t) + (N_t - K_t) \log(1 - r_m(c_t)) - \beta_1 \log(M \alpha_m^{\text{rel}}(c_t)) \right),\end{aligned} \tag{6}$$

where $\boldsymbol{\alpha}^{\text{prior}}(c_t) = [\frac{1}{M}, ..., \frac{1}{M}]^\top$ is the uniform prior belief, and β_1 is the temperature coefficient. Note that the constant term $\log \binom{N_t}{K_t}$ is omitted in Equation 6 since it does not affect optimization. To further justify the objective in Equation 6, we characterize its optimal solution and establish its connection to Bayesian inference in Proposition 4.1.

Proposition 4.1. *The optimal solution of Equation 6 recovers the following Bayesian posterior:*

$$\alpha_m^{\text{rel}}(c_t) = \frac{\alpha_m^{\text{prior}}(c_t) \exp(\ell_{t,m}/\beta_1)}{\sum_{j=1}^{M} \alpha_j^{\text{prior}}(c_t) \exp(\ell_{t,j}/\beta_1)} \propto \alpha_m^{\text{prior}}(c_t) \cdot p(K_t|N_t, c_t, \theta_m)^{1/\beta_1}, \quad m = 1, 2, ..., M \tag{7}$$

where $\alpha_m^{\text{prior}}(c_t)$ is the m-th element of $\boldsymbol{\alpha}^{\text{prior}}(c_t)$. Under a uniform belief $\boldsymbol{\alpha}^{\text{prior}}(c_t)$, we have $\alpha_m^{\text{rel}}(c_t) \propto p(K_t|N_t, c_t, \theta_m)^{1/\beta_1}$.

Remark. Proposition 4.1 provides a Bayesian interpretation of Equation 6. Specifically, optimizing Equation 6 corresponds to updating the prior belief $\boldsymbol{\alpha}^{\text{prior}}(c_t)$ according to the reliability likelihood of the supervision data in $\mathcal{D}_{\text{val}}$, thus addressing PRM uncertainty. β_1 controls the strength of this belief update: when $\beta_1 = 1$, the solution recovers the standard Bayesian posterior, while $\beta_1 > 1$ yields a flatter tempered posterior and $\beta_1 < 1$ yields a sharper one. This update directly follows Bayes' rule in Equation 2. The key difference is that, the belief network amortizes this Bayesian update, enabling the reliability-aware belief to generalize to unseen prefixes at test time.

Conservatism-Aware Belief Calibration. Although the reliability-aware belief addresses PRM uncertainty, reward overestimation may still occur since policy uncertainty is not accounted for. Therefore, we introduce a second stage of belief updating, termed *conservatism-aware belief calibration*. In this stage, we take the learned reliability-aware belief as the prior belief and further calibrate it by shifting belief mass toward reward hypotheses with lower reward predictions.

Specifically, we treat the reliability-aware belief $\boldsymbol{\alpha}^{\text{rel}}(c_t)$ as the prior belief, and define the hybrid Bayesian belief as

$$\boldsymbol{\alpha}^{\text{hyb}}(c_t) = \arg \min_{\boldsymbol{\alpha} \in \Delta^{M-1}} \left\{ \boldsymbol{\alpha}^\top \boldsymbol{r}(c_t) + \beta_2 \cdot D_{\text{KL}} \left(\boldsymbol{\alpha} \| \boldsymbol{\alpha}^{\text{rel}}(c_t) \right) \right\}, \tag{8}$$

where $\Delta^{M-1} = \{\boldsymbol{\alpha} \in \mathbb{R}^M_{\geq 0} : \sum_{m=1}^{M} \alpha_m = 1\}$ denotes the probability simplex, and $\beta_2 > 0$ is a temperature coefficient. Solving Equation 8 yields the following closed-form solution:

$$\alpha_m^{\text{hyb}}(c_t) = \frac{\alpha_m^{\text{rel}}(c_t) \exp(-r_m(c_t)/\beta_2)}{\sum_{j=1}^{M} \alpha_j^{\text{rel}}(c_t) \exp(-r_j(c_t)/\beta_2)} \propto \alpha_m^{\text{rel}}(c_t) \cdot \exp(-r_m(c_t)/\beta_2), \quad m = 1, 2, ..., M. \tag{9}$$

Remark. Equation 9 admits a Bayesian interpretation: $\boldsymbol{\alpha}^{\text{hyb}}(c_t)$ is the Bayesian posterior obtained by taking $\boldsymbol{\alpha}^{\text{rel}}(c_t)$ as the prior and $\exp(-\boldsymbol{r}(c_t)/\beta_2)$ as the likelihood. This formulation is closely related to the control-as-inference framework (Levine, 2018), where an exponentially weighted reward is used as a likelihood. The key difference is that control-as-inference typically assigns larger likelihood to higher-reward outcomes, whereas Equation 9 introduces conservatism by assigning larger likelihood to lower-reward predictions, thereby mitigating reward overestimation. Note that, unlike reliability-aware belief updating, this stage does not require training a belief network. At test time, the ensemble reward predictions $\boldsymbol{r}(c_t)$ are available. Therefore, $\boldsymbol{\alpha}^{\text{hyb}}(c_t)$ can be directly obtained by applying exponential reweighting in Equation 9 to $\boldsymbol{\alpha}^{\text{rel}}(c_t)$.

Substituting Equation 7 into Equation 9, we obtain:

$$\underbrace{\alpha_m^{\text{hyb}}(c_t)}_{\text{posterior}} \propto \underbrace{\alpha_m^{\text{prior}}(c_t)}_{\text{prior}} \cdot \underbrace{\exp(\ell_{t,m}/\beta_1 - r_m(c_t)/\beta_2)}_{\text{likelihood}}, \quad m = 1, 2, ..., M. \tag{10}$$

In Equation 10, the likelihood consists of two components: the first term, $\exp(\ell_{t,m}/\beta_1)$, accounts for the reliability of each reward hypothesis, while the second term, $\exp(-r_m(c_t)/\beta_2)$, introduces conservatism by favoring hypotheses with lower reward predictions. β_1 and β_2 control the strengths of the reliability-aware and conservatism-aware belief updates, respectively. Therefore, $\boldsymbol{\alpha}^{\text{hyb}}(c_t)$ represents a reliability- and conservatism-aware Bayesian belief. The final process reward is computed as: $\tilde{r}(c_t) = \left[\boldsymbol{\alpha}^{\text{hyb}}(c_t)\right]^\top \boldsymbol{r}(c_t)$. By aggregating the ensemble predictions under $\boldsymbol{\alpha}^{\text{hyb}}(c_t)$, $\tilde{r}(c_t)$ jointly accounts for reliability and conservatism. Specifically, it assigns larger weights to reward hypotheses that better explain the supervision data while favoring more conservative hypotheses with lower reward predictions, thereby addressing both PRM uncertainty and policy uncertainty.

4.3 Connection with Continuous Hypothesis Space

Thus far, we have considered the discrete hypothesis space represented by a finite reward ensemble for computational tractability. In this part, we further show that this finite-ensemble formulation can be interpreted as an implicit Bayesian update over an underlying continuous reward hypothesis space Θ. Specifically, let $p(\theta)$ denote the prior belief over Θ, and we assume the ensemble hypotheses $\theta_1, ..., \theta_M$ are sampled independently and identically from $p(\theta)$, that is, $\theta_1, ..., \theta_M \overset{\text{i.i.d.}}{\sim} p(\theta)$. Then $\tilde{r}(c_t) = \mathbb{E}_{\theta_{1:M} \overset{\text{i.i.d.}}{\sim} p(\theta)} \sum_{m=1}^{M} \left[\alpha^{\text{hyb}}_{\theta_m}(c_t) \cdot r_{\theta_m}(c_t)\right]$. For consistency with the preceding notations, we abbreviate $\alpha^{\text{hyb}}_{\theta_m}(c_t)$ as $\alpha^{\text{hyb}}_m(c_t)$ and $r_{\theta_m}(c_t)$ as $r_m(c_t)$. Our goal is to construct a posterior belief $\tilde{p}(\theta|c_t)$ such that $\tilde{r}(c_t) = \mathbb{E}_{\theta_{1:M} \overset{\text{i.i.d.}}{\sim} p(\theta)} \sum_{m=1}^{M} \left[\alpha^{\text{hyb}}_m(c_t) \cdot r_m(c_t)\right] = \mathbb{E}_{\theta \sim \tilde{p}(\theta|c_t)} \left[r_\theta(c_t)\right]$. In other words, we aim to show that the expected weighted sum over prior hypotheses $p(\theta)$ can be equivalently represented as an expectation under a continuous posterior belief $\tilde{p}(\theta|c_t)$. According to Bayes' rule, $\tilde{p}(\theta|c_t)$ could be obtained by reweighting $p(\theta)$ with a likelihood ratio $\xi(\theta; c_t)$, that is, $\tilde{p}(\theta|c_t) = p(\theta)\xi(\theta; c_t)$. Denote $s(\theta; c_t) \triangleq \ell_{t,\theta}/\beta_1 - r_\theta(c_t)/\beta_2$, where $\ell_{t,\theta} \triangleq \log p(K_t|N_t, c_t, \theta) = \log \binom{N_t}{K_t} + K_t \log r_\theta(c_t) + (N_t - K_t) \log (1 - r_\theta(c_t))$. Then we characterize $\xi(\theta; c_t)$ as follows.

Proposition 4.2 (Likelihood Ratio). *Let $\mathcal{F}$ denote the cumulative distribution function of $s(\theta; c_t)$ with $\theta \sim p(\theta)$. Then the likelihood ratio $\xi(\theta; c_t)$ is given by:*

$$\xi(\theta; c_t) = \sum_{m=1}^{M} \bar{\alpha}_m(\theta; c_t) \frac{M!}{(m-1)!(M-m)!} \mathcal{U}_m(\mathcal{F}(s(\theta; c_t))), \tag{11}$$

where $\mathcal{U}_m(\mathcal{F}(s(\theta; c_t))) = [\mathcal{F}(s(\theta; c_t))]^{m-1}[1 - \mathcal{F}(s(\theta; c_t))]^{M-m}$, and $\{\bar{\alpha}_m(\theta; c_t)\}_{m=1}^{M}$ denotes the weighting coefficients defined in Appendix B.2.

Remark. For each $m \in \{1, ..., M\}$, $\mathcal{U}_m(\mathcal{F}(s(\theta; c_t))$ is a unimodal continuous function. This function attains its maximum value at the $\frac{m-1}{M-1}$-quantile of $s(\theta; c_t)$, that is, $\mathcal{F}(s(\theta; c_t)) = \frac{m-1}{M-1}$. Therefore, each term in Equation 11 corresponds to a continuous component concentrated around a particular quantile of $s(\theta; c_t)$. When the prior $p(\theta)$ is uniform, the posterior satisfies $\tilde{p}(\theta|c_t) \propto \xi(\theta; c_t)$, and is a weighted mixture of these continuous components. Compared with the discrete belief in Equation 5, $\tilde{p}(\theta|c_t)$ distributes the belief weights over the continuous hypothesis space Θ. This shows that our finite-ensemble formulation can be viewed as an implicit Bayesian update that reshapes the prior belief into a continuous posterior through likelihood reweighting.

4.4 Practical Implementation

In this section, we present the practical implementation of BayesianPRM, including the model architecture and the training and inference procedures.

Model Architecture. BayesianPRM consists of a reward ensemble and a belief network built on a shared LLM backbone. Given a prefix c_t, we insert a marker token <prm>, and the LLM backbone produces a final-layer hidden representation z_t at <prm>. We attach M reward heads $\{r_m\}_{m=1}^{M}$ to z_t. To prevent ensemble collapse and maintain diversity among the reward predictions, following Lee et al. (2026), we pair the reward heads with randomly initialized and fixed prior networks $\{f_m\}_{m=1}^{M}$ (Osband et al., 2018; 2023). The prediction of the m-th process reward is given by $r_m(c_t) = \sigma(g_m(z_t) + \lambda_{\text{prior}} f_m(z_t))$, where $g_m(z_t)$ is the reward logit of r_m, λ_{prior} is a hyperparameter and $\sigma(\cdot)$ is the sigmoid function. For the belief network w_ν, it takes $x_t = [z_t; \boldsymbol{r}(c_t)]$ as the input and outputs M logits, the reliability-aware belief is computed as $\boldsymbol{\alpha}^{\text{rel}}(c_t) = \text{Softmax}(w_\nu(x_t))$.

Table 2: **PRM accuracy evaluation results.** We evaluate Brier Score, Positive Brier Score, ECE, and AverageCE on four benchmarks: MathVision, the Vision Only subset of MathVerse (denoted as MathVerse-VO), MathVista, and MMStar. ↓ indicates that lower values are better. ± denotes the standard deviation across three random training seeds.

	MathVision	MathVerse-VO	MathVista	MMStar	Average
			Brier Score ↓		
Standard PRM	0.2025±0.0194	0.2274±0.0143	0.1704±0.0082	0.2357±0.0236	0.2090
BetaPRM	0.2377±0.0221	0.2489±0.0163	0.1712±0.0148	0.2427±0.0121	0.2251
DreamPRM	0.2461±0.0292	0.2498±0.0180	0.1904±0.0064	0.2342±0.0208	0.2301
BayesianPRM	**0.1411**±0.0201	**0.1848**±0.0112	**0.1579**±0.0107	**0.2085**±0.0153	**0.1731**
			Positive Brier Score ↓		
Standard PRM	0.1931±0.0314	0.2147±0.0203	0.1403±0.0149	0.2300±0.0182	0.1945
BetaPRM	0.2308±0.0226	0.2413±0.0209	0.1496±0.0157	0.2371±0.0261	0.2147
DreamPRM	0.2428±0.0251	0.2445±0.0167	0.1808±0.0189	0.2276±0.0203	0.2239
BayesianPRM	**0.1207**±0.0085	**0.1602**±0.0113	**0.1130**±0.0076	**0.1965**±0.0128	**0.1476**
			ECE ↓		
Standard PRM	0.3213±0.0263	0.2868±0.0196	0.1481±0.0075	0.2704±0.0351	0.2567
BetaPRM	0.3777±0.0411	0.3241±0.0259	0.1654±0.0107	0.2760±0.0212	0.2858
DreamPRM	0.3585±0.0399	0.3139±0.0240	0.1902±0.0137	0.2827±0.0205	0.2863
BayesianPRM	**0.2102**±0.0116	**0.2016**±0.0154	**0.1108**±0.0072	**0.2200**±0.0108	**0.1857**
			AverageCE ↓		
Standard PRM	0.2849±0.0266	0.2140±0.0160	0.1139±0.0100	0.2159±0.0159	0.2072
BetaPRM	0.2823±0.0126	0.2401±0.0228	0.1372±0.0060	0.1901±0.0084	0.2124
DreamPRM	0.2973±0.0204	0.2305±0.0172	0.1622±0.0120	0.2384±0.0097	0.2321
BayesianPRM	**0.2301**±0.0193	**0.1733**±0.0092	**0.0946**±0.0013	**0.1720**±0.0029	**0.1675**

Training Procedure. BayesianPRM includes a two-stage training procedure. At the first stage, we train the reward ensemble on the training set $\mathcal{D}_{\text{train}}$. Specifically, the LLM backbone and each reward head are optimized with the binary cross-entropy loss in Equation 1. At the second stage, we freeze the LLM backbone and all reward heads and train only the belief network on the validation set $\mathcal{D}_{\text{val}}$ by optimizing the ELBO in Equation 6. We use $\mathcal{D}_{\text{val}}$ instead of $\mathcal{D}_{\text{train}}$ for belief network training to prevent it from overfitting to the training data used to optimize the reward ensemble.

Inference Procedure. During inference, given a prefix c_t, the reward ensemble first produces the reward predictions $\boldsymbol{r}(c_t)$, while the belief network outputs the reliability-aware belief $\boldsymbol{\alpha}^{\text{rel}}(c_t)$. We then perform conservatism-aware belief calibration to obtain $\boldsymbol{\alpha}^{\text{hyb}}(c_t)$ by Equation 9. The final process reward is computed as $\tilde{r}(c_t) = \left[\boldsymbol{\alpha}^{\text{hyb}}(c_t)\right]^{\top} \boldsymbol{r}(c_t)$. We present more implementation details and the pseudo-code of BayesianPRM in Appendix D.

5 Experiments

In this section, we empirically evaluate BayesianPRM from two perspectives. First, we investigate whether BayesianPRM improves the accuracy of process reward prediction. Second, we examine whether BayesianPRM can enhance test-time scaling performance.

5.1 Process Reward Accuracy Evaluation

Datasets. We mainly evaluate our method on multimodal reasoning benchmarks. The dataset we use during training is VisualPRM400K-v1.1 (Wang et al., 2026). For BayesianPRM, we randomly sample 80% of the data as $\mathcal{D}_{\text{train}}$ and use the remaining 20% as $\mathcal{D}_{\text{val}}$ for belief network training. For the other baselines, we use the full dataset for training. For evaluation benchmarks, we use MathVision (Wang et al., 2024b), the Vision Only subset of MathVerse (Zhang et al., 2024a) (denoted as MathVerse-VO), MathVista (Lu et al., 2024), and MMStar (Chen et al., 2024a). For each image-

Table 3: **PRM-guided test-time scaling performance comparison.** We compare Best-of-N ($N = 16$) and Instance-Adaptive Scaling (IAS, $N_{\max} = 16$) on MathVision, MathVerse-VO, MathVista and MMStar. ↑ indicates that higher values are better. ± denotes the standard deviation across three random training seeds.

	MathVision	MathVerse-VO	MathVista	MMStar	Average
InternVL3-8B	29.93	45.05	65.90	66.27	51.79
Best-of-N ($N = 16$)↑					
Self-Consistency	33.22	47.46	68.00	68.60	54.32
Standard PRM	32.57±0.78	50.78±1.17	69.00±1.10	67.47±0.61	54.96
BetaPRM	34.31±0.70	**51.85**±0.63	68.66±1.15	67.77±1.54	55.65
DreamPRM	34.79±1.01	51.22±0.64	69.13±0.65	67.18±1.12	55.58
BayesianPRM	**35.86**±0.70	51.40±1.11	**69.40**±0.53	**69.33**±0.60	**56.50**
IAS ($N_{\max} = 16$) ↑					
Standard PRM	32.19±0.68	50.46±0.61	**69.11**±0.98	67.96±0.44	54.93
BetaPRM	34.26±0.70	50.58±1.06	68.66±1.15	68.18±0.56	55.42
DreamPRM	34.57±0.63	50.26±1.12	68.64±1.06	67.93±0.65	55.35
BayesianPRM	**35.68**±0.58	**50.67**±0.60	68.92±0.51	**69.00**±0.64	**56.07**

question pair, we first use InternVL3-8B to generate a reasoning response, from which intermediate reasoning prefixes are extracted. Starting from each prefix c_t, we perform 16 Monte-Carlo rollouts using InternVL3-8B. Each rollout is then evaluated by Qwen2.5-32B-Instruct as the judge model to determine its correctness. The proportion of correct rollouts is treated as the empirical success probability $\hat{p}_t$. The $(c_t, \hat{p}_t)$ pairs constitute $\mathcal{D}_{\text{test}}$ for evaluation.

Evaluation Metrics. Following Park et al. (2026), we assess the accuracy of PRM predictions using the following calibration metrics: Brier Score (Glenn et al., 1950), Positive Brier Score (Park et al., 2026), Expected Calibration Error (ECE) (Naeini et al., 2015) and Average Calibration Error (AverageCE) (Neumann et al., 2018) between the empirical success probability and the PRM predictions. The computation of these metrics is presented in Appendix C.2.

Baselines. We consider the following baselines for comparison with BayesianPRM: (1) Standard PRM, which is optimized using the standard cross-entropy loss; (2) BetaPRM (Li et al., 2026b), which models the process reward as a Beta distribution; and (3) DreamPRM (Cao et al., 2026), which reweights different domains of the dataset for multimodal reasoning. All PRMs are built on InternVL3-8B. Since QR-PRM (Park et al., 2026) requires an additional calibration set, we provide a separate comparison with it in Appendix E.3.

Results. As shown in Table 2, BayesianPRM consistently achieves the lowest calibration error across all four benchmarks and all four metrics. Specifically, BayesianPRM reduces the average Brier Score, Positive Brier Score, ECE, and AverageCE by **17.2%**, **24.1%**, **27.7%**, and **19.2%**, respectively, compared with Standard PRM. In particular, BayesianPRM achieves a substantially lower Positive Brier Score, suggesting that conservatism-aware belief calibration effectively reduces reward overestimation. Meanwhile, BayesianPRM also consistently improves calibration on benchmarks where Standard PRM already exhibits relatively small prediction errors, such as MathVista, suggesting that reliability-aware belief updating provides benefits beyond simply mitigating systematic overestimation.

5.2 Test-time Scaling Performance

Experimental Setup. We use the same $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$ as in Section 5.1. The test dataset $\mathcal{D}_{\text{test}}$ consists of the image-question pairs from the evaluation benchmarks. We compare the performance of Best-of-N and Instance-Adaptive Scaling (IAS) (Park et al., 2026). For Best-of-N, we set $N = 16$. For each sample in $\mathcal{D}_{\text{test}}$, InternVL3-8B generates 16 candidate responses, which are scored by the PRMs, and the response with the highest score is selected. For IAS, we set $N_{\max} = 16$ and dynamically allocate the rollout budget; the detailed pro-

cedure is provided in Appendix C.3. We use Standard PRM, BetaPRM, and DreamPRM as baselines. For Best-of-N, we add an additional Self-Consistency (Wang et al., 2022) baseline.

Results. As shown in Table 3, BayesianPRM enhances test-time scaling performance under both Best-of-N and IAS. Under Best-of-N with $N = 16$, BayesianPRM achieves the highest average accuracy of **56.50**, outperforming Standard PRM, BetaPRM, and DreamPRM by 1.54, 0.85, and 0.92 points, respectively. BayesianPRM obtains the best performance on MathVision, MathVista, and MMStar, while remaining competitive on MathVerse-VO, where it is only 0.45 points below BetaPRM. With IAS and $N_{\max} = 16$, BayesianPRM again achieves the highest average accuracy of **56.07**, exceeding Standard PRM, BetaPRM, and DreamPRM by 1.14, 0.65, and 0.72 points, respectively. It achieves the best result on MathVision, MathVerse-VO, and MMStar. On MathVista, BayesianPRM reaches 68.92, only 0.19 points below Standard PRM.

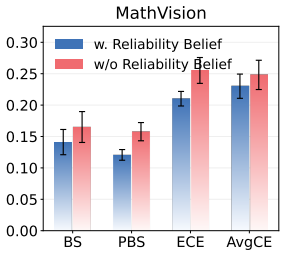


Figure 2: PRM calibration results comparison on MathVision dataset.

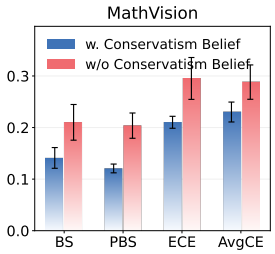


Figure 3: PRM calibration results comparison on MathVision dataset.

Table 4: Best-of-16 results with and without reliability belief.

	w. Rel.	w/o Rel.
Vision	**35.86**	35.11
Verse	**51.40**	50.87
Vista	**69.40**	68.94
Star	**69.33**	68.56

Table 5: Best-of-16 results with and without conservatism belief.

	w. Con.	w/o Con.
Vision	**35.86**	35.09
Verse	**51.40**	50.83
Vista	**69.40**	68.80
Star	**69.33**	68.22

5.3 Ablation Study

In this section, we conduct an ablation study for BayesianPRM. We mainly examine the necessity of the reliability-aware belief updating and conservatism-aware belief calibration.

Reliability-Aware Belief Updating. To investigate the necessity of reliability-aware belief updating, we first train the reward ensemble using the standard cross-entropy loss, and then directly reweight the uniform prior toward lower reward predictions, omitting the belief network training step. Figure 2 compares the calibration metrics on MathVision with and without reliability-aware belief updating, where BS, PBS, AvgCE denote Brier Score, Positive Brier Score, and Average Calibration Error. Removing reliability-aware belief updating increases all four metrics, indicating worse calibration and less accurate reward estimates. Table 4 compares the Best-of-16 results on four datasets, where Vision, Verse, Vista, and Star correspond to MathVision, MathVerse-VO, MathVista, and MMStar, respectively. Performance consistently drops after removing reliability-aware belief updating. Together, these results show that reliability-aware belief updating is important for both reward calibration and downstream test-time scaling.

Conservatism-Aware Belief Calibration. To examine the necessity of conservatism-aware belief calibration, we directly use the reliability-aware belief for reward estimation. Figure 3 and Table 5 report the calibration results on MathVision and the test-time scaling results across four benchmarks, respectively. It is clear that removing conservatism-aware belief calibration consistently degrades both calibration and test-time scaling performance, indicating the importance of this step. More ablation study results are deferred to Appendix E.4 due to space limit.

6 Conclusion

In this work, we investigate the uncertainty in process reward modeling and identify two main sources: PRM uncertainty and policy uncertainty. To jointly address both uncertainties, we propose BayesianPRM, which formulates process reward modeling as Bayesian learning and consists of two key steps: reliability-aware belief updating and conservatism-aware belief calibration. Experimental results across various multimodal reasoning benchmarks indicate that BayesianPRM consistently improves both PRM reward prediction and test-time scaling performance.

7 AI USE STATEMENT

In this work, we use LLMs to improve the grammar and presentation of the manuscript and to assist with parts of the experimental code. We affirm that LLMs were not involved in the core research contributions, including the conceptualization of the idea, the development of methodology, and theoretical proofs. All LLM-assisted content and code were reviewed and verified by the authors. The authors take full responsibility for the final content of this work.

8 ETHICS STATEMENT

This research does not involve human subjects, animal experiments, or sensitive datasets, nor does it include applications that may pose ethical risks. We believe that this study does not raise any ethical concerns.

9 REPRODUCIBILITY STATEMENT

We have made efforts to ensure that our work is reproducible. We provide the training scripts of BayesianPRM in the supplementary materials. Additionally, we provide the detailed experimental setup in Appendix C, and the implementation details in Appendix D.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, 2024.

Neil Band, Xuechen Li, Tengyu Ma, and Tatsunori Hashimoto. Linguistic calibration of long-form generations. *arXiv preprint arXiv:2404.00474*, 2024.

Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. Forest-of-thought: Scaling test-time compute for enhancing llm reasoning. *arXiv preprint arXiv:2412.09078*, 2024.

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.

Charlotte Bunne, Andreas Krause, and Marco Cuturi. Supervised training of conditional monge maps. *Advances in Neural Information Processing Systems*, 35:6859–6872, 2022.

Qi Cao, Ruiyi Wang, Ruiyi Zhang, Sai Ashish Somayajula, and Pengtao Xie. Dreamprm: Domain-reweighted process reward model for multimodal reasoning. *Advances in Neural Information Processing Systems*, 38:122199–122226, 2026.

Hyungjoo Chae, Seonghwan Kim, Junhee Cho, Seungone Kim, Seungjun Moon, Gyeom Hwangbo, Dongha Lim, Minjin Kim, Yeonjun Hwang, Minju Gwak, et al. Web-shepherd: Advancing prms for reinforcing web agents. *Advances in Neural Information Processing Systems*, 38:63314–63356, 2026.

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.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, et al. Are we on the right way for evaluating large vision-language models? *Advances in Neural Information Processing Systems*, 37:27056–27087, 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.

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.

Thomas Coste, Usman Anwar, Robert Kirk, and David Krueger. Reward model ensembles help mitigate overoptimization. In *International Conference on Learning Representations*, volume 2024, pp. 50905–50931, 2024.

Xibin Dong, Zhiwen Yu, Wenming Cao, Yifan Shi, and Qianli Ma. A survey on ensemble learning. *Frontiers of Computer Science*, 14(2):241–258, 2020.

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.

Jacob Eisenstein, Chirag Nagpal, Alekh Agarwal, Ahmad Beirami, Alex D'Amour, DJ Dvijotham, Adam Fisch, Katherine Heller, Stephen Pfohl, Deepak Ramachandran, et al. Helping or herding? reward model ensembles mitigate but do not eliminate reward hacking. *arXiv preprint arXiv:2312.09244*, 2023.

Ekaterina Fadeeva, Aleksandr Rubashevskii, Artem Shelmanov, Sergey Petrakov, Haonan Li, Hamdy Mubarak, Evgenii Tsymbalov, Gleb Kuzmin, Alexander Panchenko, Timothy Baldwin, et al. Fact-checking the output of large language models via token-level uncertainty quantification. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 9367–9385, 2024.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International conference on machine learning*, pp. 10835–10866. PMLR, 2023.

W Brier Glenn et al. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Nan Duan, Weizhu Chen, et al. Critic: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, volume 2024, pp. 57734–57811, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Junyu Guo, Shangding Gu, Ming Jin, Costas Spanos, and Javad Lavaei. Llms should express uncertainty explicitly. *arXiv preprint arXiv:2604.05306*, 2026.

Jiuzhou Han, Wray Buntine, and Ehsan Shareghi. Towards uncertainty-aware language agent. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 6662–6685, 2024.

Jiuzhou Han, Wray Buntine, and Ehsan Shareghi. Uncertainty-based methods for automated process reward data construction and output aggregation in mathematical reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 30925–30933, 2026.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Overview of supervised learning. In *The elements of statistical learning: Data mining, inference, and prediction*, pp. 9–41. Springer, 2008.

Jian Hu et al. Reinforce++: A simple and efficient approach for aligning large language models. *arXiv preprint arXiv:2501.03262*, 1(4), 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.

Robert Irvine, Douglas Boubert, Vyas Raina, Adian Liusie, Ziyi Zhu, Vineet Mudupalli, Aliaksei Korshuk, Zongyi Liu, Fritz Cremer, Valentin Assassi, et al. Rewarding chatbots for real-world engagement with millions of users. *arXiv preprint arXiv:2303.06135*, 2023.

Minki Kang, Jongwon Jeong, and Jaewoong Cho. T1: Tool-integrated verification for test-time compute scaling in small language models. In *The Fourteenth International Conference on Learning Representations*, 2026.

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.

Roger Koenker and Kevin F Hallock. Quantile regression. *Journal of economic perspectives*, 15(4): 143–156, 2001.

Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. *Advances in neural information processing systems*, 7, 1994.

Kuang-Huei Lee, Ian Fischer, Yueh-Hua Wu, Dave Marwood, Shumeet Baluja, Dale Schuurmans, and Xinyun Chen. Evolving deeper llm thinking. *arXiv preprint arXiv:2501.09891*, 2025.

Yoonho Lee, Jonathan Williams, Henrik Marklund, Archit Sharma, Eric Mitchell, Anikait Singh, and Chelsea Finn. Inference-time alignment via hypothesis reweighting. *Transactions on Machine Learning Research*, 2026.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

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*, 2026a.

Jinyuan Li, Langlin Huang, Chengsong Huang, Shaoyang Xu, Donghong Cai, Yuyi Yang, Wenxuan Zhang, and Jiaxin Huang. Process rewards with learned reliability. *arXiv preprint arXiv:2605.15529*, 2026b.

Xiaobo Liang, Haoke Zhang, Juntao Li, Kehai Chen, Qiaoming Zhu, and Min Zhang. Generative reward modeling via synthetic criteria preference learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26755–26769, 2025.

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 *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Stephanie Lin, Jacob Hilton, and Owain Evans. Teaching models to express their uncertainty in words. *arXiv preprint arXiv:2205.14334*, 2022.

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. *arXiv preprint arXiv:2502.06703*, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

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 *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, 2021.

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 *International Conference on Learning Representations*, volume 2024, pp. 23439–23554, 2024.

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.

Rachel Ma, Dylan Hadfield-Menell, and Kristjan Greenewald. Distributional process reward models: Calibrated prediction of future rewards via conditional optimal transport. *arXiv preprint arXiv:2605.06785*, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Andrey Malinin and Mark Gales. Uncertainty estimation in autoregressive structured prediction. *arXiv preprint arXiv:2002.07650*, 2020.

Potsawee Manakul, Adian Liusie, and Mark Gales. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 9004–9017, 2023.

Sabrina J Mielke, Arthur Szlam, Emily Dinan, and Y-Lan Boureau. Reducing conversational agents' overconfidence through linguistic calibration. *Transactions of the Association for Computational Linguistics*, 10:857–872, 2022.

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Lukas Neumann, Andrew Zisserman, and Andrea Vedaldi. Relaxed softmax: Efficient confidence auto-calibration for safe pedestrian detection. 2018.

Alex Nguyen, Dheeraj Mekala, Chengyu Dong, and Jingbo Shang. When is the consistent prediction likely to be a correct prediction? *arXiv preprint arXiv:2407.05778*, 2024.

Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

Ian Osband, Zheng Wen, Seyed Mohammad Asghari, Vikranth Dwaracherla, Morteza Ibrahimi, Xiuyuan Lu, and Benjamin Van Roy. Epistemic neural networks. *Advances in Neural Information Processing Systems*, 36:2795–2823, 2023.

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 Alimohammadi, Hao Wang, and Navid Azizan. Know what you don't know: Uncertainty calibration of process reward models. *Advances in Neural Information Processing Systems*, 38:38852–38895, 2026.

Runqi Qiao, Qiuna Tan, Guanting Dong, MinhuiWu MinhuiWu, Chong Sun, Xiaoshuai Song, Jiapeng Wang, Zhuoma Gongque, Shanglin Lei, Yifan Zhang, et al. We-math: Does your large multimodal model achieve human-like mathematical reasoning? In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 20023–20070, 2025.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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

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.

Ola Shorinwa, Zhiting Mei, Justin Lidard, Allen Z Ren, and Anirudha Majumdar. A survey on uncertainty quantification of large language models: Taxonomy, open research challenges, and future directions. *ACM Computing Surveys*, 58(3):1–38, 2025.

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.

Yifan Song, Guoyin Wang, Sujian Li, and Bill Yuchen Lin. The good, the bad, and the greedy: Evaluation of llms should not ignore non-determinism. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4195–4206, 2025.

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.

El-Ghazali Talbi. A taxonomy of metaheuristics for bi-level optimization. In *Metaheuristics for bi-level optimization*, pp. 1–39. Springer, 2013.

Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

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.

Evan Wang, Federico Cassano, Catherine Wu, Yunfeng Bai, William Song, Vaskar Nath, Ziwen Han, Sean Hendryx, Summer Yue, and Hugh Zhang. Planning in natural language improves llm search for code generation. In *International Conference on Learning Representations*, volume 2025, pp. 2432–2478, 2025.

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, 2024a.

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, 2024b.

Weiyun Wang, Zhangwei Gao, Lianjie Chen, Zhe Chen, Jinguo Zhu, Xiangyu Zhao, Yangzhou Liu, Yue Cao, Shenglong Ye, Xizhou Zhu, et al. Visualprm400k: An effective dataset for training multimodal process reward models. In *The Fourteenth International Conference on Learning Representations*, 2026.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Yuyang Wu, Yifei Wang, Ziyu Ye, Tianqi Du, Stefanie Jegelka, and Yisen Wang. When more is less: Understanding chain-of-thought length in llms. In *International Conference on Learning Representations*, volume 2026, pp. 127108–127138, 2026.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Runyang You, Yongqi Li, Meng Liu, Wenjie Wang, Liqiang Nie, and Wenjie Li. Parallel test-time scaling for latent reasoning models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 44703–44717, 2026.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9556–9567, 2024.

Jianghangfan Zhang, Yibo Yan, Kening Zheng, Xin Zou, Song Dai, and Xuming Hu. Gm-prm: A generative multimodal process reward model for multimodal mathematical reasoning. In *Proceedings of the 4th Workshop on Advances in Language and Vision Research (ALVR)*, pp. 139–154, 2026.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, et al. A survey on test-time scaling in large language models: What, how, where, and how well? *arXiv preprint arXiv:2503.24235*, 2025.

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, 2024a.

Yunxiang Zhang, Muhammad Khalifa, Lajanugen Logeswaran, Jaekyeom Kim, Moontae Lee, Honglak Lee, and Lu Wang. Small language models need strong verifiers to self-correct reasoning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 15637–15653, 2024b.

Jian Zhao, Runze Liu, Kaiyan Zhang, Zhimu Zhou, Junqi Gao, Dong Li, Jiafei Lyu, Zhouyi Qian, Biqing Qi, Xiu Li, et al. Genprm: Scaling test-time compute of process reward models via generative reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 34932–34940, 2026.

Congmin Zheng, Jiachen Zhu, Zhuoying Ou, Yuxiang Chen, Kangning Zhang, Rong Shan, Zeyu Zheng, Mengyue Yang, Jianghao Lin, Yong Yu, et al. A survey of process reward models: From outcome signals to process supervisions for large language models. *arXiv preprint arXiv:2510.08049*, 2025.

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.

Chengke Zou, Xingang Guo, Rui Yang, Junyu Zhang, Bin Hu, and Huan Zhang. Dynamath: A dynamic visual benchmark for evaluating mathematical reasoning robustness of vision language models. In *International Conference on Learning Representations*, volume 2025, pp. 48337–48383, 2025.

A Related Work

Process Reward Models. Reward models play a crucial role in reinforcement learning (RL) (Schulman et al., 2017; Ahmadian et al., 2024; Hu et al., 2025; Guo et al., 2025) and test-time scaling (Snell et al., 2024; Huang et al., 2025; Chae et al., 2026; You et al., 2026) for reasoning tasks. Process reward models (PRMs) (Lightman et al., 2024; Du et al., 2025; Sun et al., 2025; Zheng et al., 2025; Pala et al., 2025) provide step-level rewards for intermediate reasoning steps, and have been shown more effective than outcome reward models (ORMs) (Uesato et al., 2022; Cobbe et al., 2021; Shao et al., 2024). PRMs can be categorized into value-based PRMs (Li et al., 2026b;a; Park et al., 2026) and generative PRMs (Zhao et al., 2026; Khalifa et al., 2025; Liang et al., 2025; Zhang et al., 2026). Our work falls into the scope of value-based PRMs, where PRMs are required to predict the process reward that reflects the success rate of a given prefix estimated by Monte-Carlo sampling (Wang et al., 2024b; Luo et al., 2024). However, due to the inherent limitations of supervised learning (Hastie et al., 2008), value-based PRMs may produce inaccurate process rewards. To address this, recent works (Li et al., 2026b; Park et al., 2026; Ma et al., 2026) have modeled PRMs as distributions rather than point estimates. BayesianPRM can also be seen as a distributional PRM, where the posterior belief represents the probability of different reward hypotheses, and thus induces a distribution over possible process reward values.

Test-Time Scaling. Test-time scaling (Snell et al., 2024; Brown et al., 2024; Gou et al., 2024; Zhang et al., 2025) improves reasoning quality by allocating additional computation at inference time. Representative methods include *parallel scaling* such as self-consistency (Brown et al., 2024; Irvine et al., 2023; Song et al., 2025; Nguyen et al., 2024; Wu et al., 2026), PlanSearch (Wang et al., 2025), and *sequential scaling* such as self-refine (Zhang et al., 2024b; Madaan et al., 2023), sequential revision (Lee et al., 2025), ReAct (Yao et al., 2022), and so on. A common and simple method is Best-of-N (Lightman et al., 2024), which selects the best solution among multiple candidates evaluated by a verifier or reward model. Some works have also explored to adaptively allocate the compute budget to improve computational efficiency, such as Instance-Adaptive Scaling (IAS) (Park et al., 2026) and Adaptive Computation Allocation (ACA) (Li et al., 2026b).

Uncertainty Quantification for LLMs. Uncertainty quantification plays a crucial role in assessing the confidence of LLM outputs (Shorinwa et al., 2025). Prior studies (Han et al., 2024; 2026) have shown a strong correlation between the uncertainty and correctness of LLM responses. Major uncertainty quantification methods for LLMs include (1) *white-box methods*, which quantify the uncertainty with visible metrics such as entropy (Malinin & Gales, 2020), perplexity (Fadeeva et al., 2024), token log-likelihood (Manakul et al., 2023), and so on; (2) *black-box methods*, which quantify the uncertainty entirely from the LLM responses to a given prompt. Representative methods include self-consistency (Brown et al., 2024), self-verbalized methods (Band et al., 2024; Lin et al., 2022; Mielke et al., 2022; Guo et al., 2026) that train the LLM to provide a natural-language estimate of its confidence, and so on. While these studies primarily focus on quantifying the uncertainty of LLM outputs, our work explores to capture the uncertainty of PRMs with Bayesian learning.

Ensemble Reward Models. Ensemble learning (Dong et al., 2020; Krogh & Vedelsby, 1994) is a classical machine learning approach that improves predictive performance and uncertainty estimation for neural networks. Previous studies (Coste et al., 2024; Eisenstein et al., 2023) have shown that ensemble reward models could effectively mitigate reward over-optimization (Gao et al., 2023). Among all prior works, HyRe (Lee et al., 2026) and ActivePRM (Duan et al., 2025) are the closest to our work. HyRe adjusts the weights of reward ensemble at test time to adapt to different preferences, while ActivePRM leverages ensemble disagreement for active learning. In contrast, BayesianPRM captures uncertainty through Bayesian belief updating at both training and test time, distinguishing our approach from prior methods.

B Proofs

B.1 Proof of Proposition 4.1

Proof. Since the objective in Equation 6 is separable across different prefixes, we can optimize the reliability-aware belief for each prefix c_t independently. For a fixed c_t, minimizing Equation 6 is

equivalent to solving

$$\boldsymbol{\alpha}^{\text{rel}}(c_t) = \arg\max_{\boldsymbol{\alpha}\in\Delta^{M-1}} \left\{ \sum_{m=1}^{M} \alpha_m \ell_{t,m} - \beta_1 D_{\text{KL}}\left(\boldsymbol{\alpha} \| \boldsymbol{\alpha}^{\text{prior}}(c_t)\right) \right\},$$

where $\Delta^{M-1} = \{\boldsymbol{\alpha} \in \mathbb{R}^M_{\geq 0} : \sum_{m=1}^{M} \alpha_m = 1\}$ denotes the probability simplex. Expanding the KL divergence gives

$$\boldsymbol{\alpha}^{\text{rel}}(c_t) = \arg\max_{\boldsymbol{\alpha}\in\Delta^{M-1}} \left\{ \sum_{m=1}^{M} \alpha_m \ell_{t,m} - \beta_1 \sum_{m=1}^{M} \alpha_m \log \frac{\alpha_m}{\alpha_m^{\text{prior}}(c_t)} \right\}.$$

Introducing a Lagrange multiplier λ for the constraint $\sum_{m=1}^{M} \alpha_m = 1$ and multipliers $\{\mu_m\}_{m=1}^{M}$ for the constraints $\alpha_m \geq 0$, the Lagrangian function is

$$\mathcal{J}(\boldsymbol{\alpha}, \lambda, \boldsymbol{\mu}) = \sum_{m=1}^{M} \alpha_m \ell_{t,m} - \beta_1 \sum_{m=1}^{M} \alpha_m \log \frac{\alpha_m}{\alpha_m^{\text{prior}}(c_t)} + \lambda \left(\sum_{m=1}^{M} \alpha_m - 1 \right) + \sum_{m=1}^{M} \mu_m \alpha_m,$$

where $\mu_m \geq 0$ for all m. Due to the presence of the KL term, the optimal solution lies in the interior of the probability simplex, i.e., $\alpha_m > 0$ for every m. By complementary slackness, $\mu_m \alpha_m = 0$, which immediately implies $\mu_m = 0$. Taking the derivative of the Lagrangian function and setting it to zero yields:

$$\ell_{t,m} - \beta_1 \left(\log \frac{\alpha_m}{\alpha_m^{\text{prior}}(c_t)} + 1 \right) + \lambda = 0.$$

Therefore,

$$\alpha_m = \alpha_m^{\text{prior}}(c_t) \exp\left(\frac{\ell_{t,m}}{\beta_1}\right) \exp\left(\frac{\lambda - \beta_1}{\beta_1}\right).$$

The last exponential term is shared by all reward hypotheses and can be determined by the normalization constraint. Enforcing $\sum_{m=1}^{M} \alpha_m = 1$ gives

$$\alpha_m^{\text{rel}\star}(c_t) = \alpha_m(c_t) = \frac{\alpha_m^{\text{prior}}(c_t) \exp\left(\ell_{t,m}/\beta_1\right)}{\sum_{j=1}^{M} \alpha_j^{\text{prior}}(c_t) \exp\left(\ell_{t,j}/\beta_1\right)}.$$

Since

$$\ell_{t,m} = \log p(K_t \mid N_t, c_t, \theta_m),$$

we further obtain

$$\alpha_m^{\text{rel}\star}(c_t) \propto \alpha_m^{\text{prior}}(c_t) \cdot p(K_t \mid N_t, c_t, \theta_m)^{1/\beta_1}.$$

For $\beta_1 > 0$, the objective is strictly concave with respect to $\boldsymbol{\alpha}$ in the interior of the simplex, since the KL term is strictly convex. Hence, the stationary point above is the unique global optimum. In particular, when $\beta_1 = 1$, we recover

$$\alpha_m^{\text{rel}\star}(c_t) \propto \alpha_m^{\text{prior}}(c_t) p(K_t \mid N_t, c_t, \theta_m),$$

which is exactly the standard Bayesian posterior update.

□

B.2 Proof of Proposition 4.2

Proof. For a prefix c_t, let $s_{(1)} \leq s_{(2)} \leq \cdots \leq s_{(M)}$ denote the order statistics of $\{s(\theta_m; c_t)\}_{m=1}^{M}$, where $\theta_1, \ldots, \theta_M \overset{\text{i.i.d.}}{\sim} p(\theta)$. Let $\mathcal{F}(s)$ denote the cumulative distribution function of $s(\theta; c_t)$ induced by the prior $p(\theta)$, respectively.

According to the standard distribution of order statistics, the density of a hypothesis associated with the m-th ordered score can be obtained by reweighting the prior hypothesis density according to the probability that its score occupies the m-th rank:

$$p_{(m)}(\theta) = \frac{M!}{(m-1)!(M-m)!} [\mathcal{F}(s(\theta; c_t))]^{m-1} [1 - \mathcal{F}(s(\theta; c_t))]^{M-m} p(\theta). \tag{12}$$

Let $\theta_{(m)}$ and $\alpha^{\text{hyb}}_{(m)}(c_t)$ denote the reward hypothesis and hybrid belief associated with $s_{(m)}$, respectively. We define the weighting coefficients $\{\bar{\alpha}_m(\theta; c_t)\}_{m=1}^M$ as

$$\bar{\alpha}_m(\theta; c_t) = \mathbb{E}_{\theta_{1:M} \sim p(\theta)} \left[\alpha^{\text{hyb}}_{(m)}(c_t) \mid \theta_{(m)} = \theta \right], \qquad m = 1, 2, ..., M.$$

Intuitively, $\bar{\alpha}_m(\theta; c_t)$ measures the expected belief weight received by a hypothesis θ when it occupies the m-th rank under the score function $s(\theta; c_t)$. Recall that our goal is to construct a posterior belief $\tilde{p}(\theta|c_t)$ such that the expected reward under this posterior is equivalent to the weighted expectation over the reward hypotheses sampled from the prior $p(\theta)$. We first consider the latter:

$$\begin{aligned}
\mathbb{E}_{\theta_{1:M} \overset{\text{i.i.d.}}{\sim} p(\theta)} \left[\sum_{m=1}^M \alpha^{\text{hyb}}_m(c_t) r_m(c_t) \right] &= \mathbb{E}_{\theta_{1:M} \overset{\text{i.i.d.}}{\sim} p(\theta)} \left[\sum_{m=1}^M \alpha^{\text{hyb}}_{(m)}(c_t) r_{\theta_{(m)}}(c_t) \right] \\
&= \sum_{m=1}^M \mathbb{E}_{\theta_{1:M} \overset{\text{i.i.d.}}{\sim} p(\theta)} \left[\alpha^{\text{hyb}}_{(m)}(c_t) r_{\theta_{(m)}}(c_t) \right] \\
&= \sum_{m=1}^M \mathbb{E}_{\theta_{(m)}} \left[r_{\theta_{(m)}} \mathbb{E}_{\theta_{1:M} \overset{\text{i.i.d.}}{\sim} p(\theta)} \left[\alpha^{\text{hyb}}_{(m)}(c_t) \mid \theta_{(m)} \right] \right] \\
&= \sum_{m=1}^M \mathbb{E}_{\theta_{(m)}} \left[\bar{\alpha}_m(\theta_{(m)}; c_t) r_{\theta_{(m)}} \right].
\end{aligned} \tag{13}$$

The third equation holds by the law of total expectation. According to the probability density of the m-th order statistic in Equation 12, we have

$$\begin{aligned}
&\sum_{m=1}^M \mathbb{E}_{\theta_{(m)}} \left[\bar{\alpha}_m(\theta_{(m)}; c_t) r_{\theta_{(m)}} \right] = \sum_{m=1}^M \int_\Theta r_\theta(c_t) \bar{\alpha}_m(\theta; c_t) p_{(m)}(\theta) d\theta \\
&= \sum_{m=1}^M \int_\Theta r_\theta(c_t) \bar{\alpha}_m(\theta; c_t) \frac{M!}{(m-1)!(M-m)!} \cdot [\mathcal{F}(s(\theta; c_t))]^{m-1} [1 - \mathcal{F}(s(\theta; c_t))]^{M-m} p(\theta) d\theta \\
&= \int_\Theta r_\theta(c_t) \sum_{m=1}^M \bar{\alpha}_m(\theta; c_t) \frac{M!}{(m-1)!(M-m)!} \cdot [\mathcal{F}(s(\theta; c_t))]^{m-1} [1 - \mathcal{F}(s(\theta; c_t))]^{M-m} p(\theta) d\theta.
\end{aligned} \tag{14}$$

On the other hand, the posterior belief satisfies $\tilde{p}(\theta|c_t) = p(\theta)\xi(\theta; c_t)$, and therefore,

$$\mathbb{E}_{\theta \sim \tilde{p}(\theta|c_t)}[r_\theta(c_t)] = \int_\Theta r_\theta(c_t) \xi(\theta; c_t) p(\theta) d\theta. \tag{15}$$

By comparing the two integral forms in Equation 14 and Equation 15, we obtain

$$\xi(\theta; c_t) = \sum_{m=1}^M \bar{\alpha}_m(\theta; c_t) \frac{M!}{(m-1)!(M-m)!} [\mathcal{F}(s(\theta; c_t))]^{m-1} [1 - \mathcal{F}(s(\theta; c_t))]^{M-m},$$

which is exactly the likelihood ratio stated in Proposition 4.2.

The last step is to verify that $\tilde{p}(\theta|c_t) = p(\theta)\xi(\theta; c_t)$ is a valid probability distribution, that is, $\int_\Theta p(\theta)\xi(\theta; c_t)d\theta = 1$. The derivation is presented as follows.

$$\begin{aligned}
\int_\Theta p(\theta)\xi(\theta; c_t)d\theta &= \sum_{m=1}^{M} \int_\Theta \bar{\alpha}_m(\theta; c_t) \cdot \frac{M!}{(m-1)!(M-m)!}[\mathcal{F}(s(\theta; c_t))]^{m-1}[1-\mathcal{F}(s(\theta; c_t))]^{M-m} p(\theta)d\theta \\
&= \sum_{m=1}^{M} \int_\Theta \bar{\alpha}_m(\theta; c_t) \cdot p_{(m)}(\theta)d\theta \\
&= \sum_{m=1}^{M} \int_\Theta \mathbb{E}_{\theta_{1:M}}\left[\alpha^{\text{hyb}}_{(m)}(c_t) \mid \theta_{(m)} = \theta\right] p_{(m)}(\theta)d\theta \\
&= \sum_{m=1}^{M} \mathbb{E}_{\theta_{(m)}}\left[\mathbb{E}_{\theta_{1:M}}\left[\alpha^{\text{hyb}}_{(m)}(c_t) \mid \theta_{(m)} = \theta\right]\right] \\
&= \sum_{m=1}^{M} \mathbb{E}_{\theta_{1:M}}[\alpha^{\text{hyb}}_{(m)}(c_t)] \\
&= \mathbb{E}_{\theta_{1:M}}\left[\sum_{m=1}^{M} \alpha^{\text{hyb}}_{(m)}(c_t)\right] \\
&= 1.
\end{aligned}$$

The fifth equation holds by the law of total expectation, and the last equation holds since $\sum_{m=1}^{M} \alpha^{\text{hyb}}_{(m)}(c_t) = 1$. Hence, $\tilde{p}(\theta|c_t) = p(\theta)\xi(\theta; c_t)$ defines a valid posterior belief, then we conclude the proof.

□

C Experimental Setup

C.1 Backbones and Datasets

Backbones. We instantiate BayesianPRM and other baselines including standard PRM, BetaPRM and DreamPRM with the same InternVL3-8B backbone across all experiments. For each method, we insert a <prm> marker after the reasoning prefix c_t, and supervise the PRMs to predict the process reward at this marker position. For BayesianPRM, we attach multiple lightweight linear heads to the marker hidden state to instantiate the reward ensemble. Compared with the base model, these additional heads introduce only a negligible number of parameters. During fine-tuning, we freeze the vision encoder while updating the language model and the added linear layers. All experiments are conducted on 4×H100 GPUs, and a complete fine-tuning run takes approximately 20 hours.

Training Datasets. Following BetaPRM, we train all PRMs on VisualPRM400k-v1.1, which is constructed using Monte-Carlo rollouts generated by the InternVL2.5 series models. We use the version "-v1.1" because it provides the success-count information required for training BayesianPRM. For each reasoning prefix, the dataset reports not only the empirical success rate but also the number of successful trajectories among 16 Monte-Carlo rollouts, which is required to compute the binomial likelihood in Equation 6. VisualPRM400K-v1.1 comprises multiple sub-datasets across diagram understanding, chart and document QA, general visual question answering, science reasoning, and mathematical/geometry reasoning. For each sub-dataset, we randomly split the data at the problem level, assigning 80% to $\mathcal{D}_{\text{train}}$ and the remaining 20% to $\mathcal{D}_{\text{val}}$. The reward ensemble is trained on $\mathcal{D}_{\text{train}}$ using the standard cross-entropy loss, while the belief network is trained on the held-out $\mathcal{D}_{\text{val}}$ according to Equation 6. We use different data for the two training stages to prevent the belief network from overfitting to samples already seen by the reward ensemble, which could lead to overly optimistic reliability estimates. Training the belief network on $\mathcal{D}_{\text{val}}$ allows it to learn reliability patterns that generalize to unseen reasoning prefixes. For the other baselines, we use the full dataset for training.

Evaluation Datasets. In Sections 5.1 and 5.2, we evaluate PRMs on MathVision, MathVerse, MathVista, and MMStar. We construct the evaluation datasets for each benchmark as follows. We first

sample a subset of image-question pairs from the original datasets and use InternVL3-8B to generate a response for each pair. Each response is segmented into reasoning steps, from which we extract reasoning prefixes at different positions along the reasoning trajectory. To control the annotation cost, we randomly sample 1,000 prefixes from the resulting pool for each benchmark. For every selected prefix, we use InternVL3-8B to generate 16 Monte-Carlo rollouts. Each rollout is judged by Qwen2.5-32B-Instruct for correctness against the ground-truth answer. We then compute the empirical success rate of each prefix as the fraction of correct rollouts among the 16 samples. Finally, the reasoning prefixes and corresponding empirical success rates constitute the evaluation datasets.

C.2 Evaluation Metrics

In this part, we introduce the calculation of the PRM accuracy evaluation metrics used in Section 5.1. Let $\hat{q}_t \in [0, 1]$ denote the process reward predicted by a PRM for the reasoning prefix c_t, and let $q_t \in [0, 1]$ denote its empirical success rate estimated from Monte-Carlo rollouts. We use the following metrics to evaluate the accuracy of PRM predictions. Lower values indicate better performance for all metrics. We use $B = 10$ bins for all bin-based calibration metrics.

Brier Score. The Brier Score measures the mean squared error between the predicted process reward and the empirical success rate:

$$\text{Brier} = \frac{1}{T}\sum_{t=1}^{T}(\hat{q}_t - q_t)^2 .$$

It measures the overall accuracy of probabilistic reward predictions, regardless of whether the prediction error corresponds to overestimation or underestimation.

Positive Brier Score. To specifically quantify reward overestimation, we further use the Positive Brier Score:

$$\text{PosBrier} = \frac{1}{T}\sum_{t=1}^{T}[\max(\hat{q}_t - q_t, 0)]^2 .$$

Different from the standard Brier Score, Positive Brier only penalizes cases where the predicted process reward exceeds the empirical success rate, and therefore directly reflects the severity of reward overestimation.

Expected Calibration Error (ECE). ECE measures the discrepancy between predicted rewards and empirical success rates after partitioning the prediction interval $[0, 1]$ into B equal-width bins. Let $\mathcal{I}_b$ denote the set of samples whose predictions fall into the b-th bin. We define

$$\text{Pred}(\mathcal{I}_b) = \frac{1}{|\mathcal{I}_b|}\sum_{t\in\mathcal{I}_b}\hat{q}_t, \qquad \text{Succ}(\mathcal{I}_b) = \frac{1}{|\mathcal{I}_b|}\sum_{t\in\mathcal{I}_b} q_t.$$

ECE is then computed as

$$\text{ECE} = \sum_{b=1}^{B}\frac{|\mathcal{I}_b|}{T}\left|\text{Pred}(\mathcal{I}_b) - \text{Succ}(\mathcal{I}_b)\right| .$$

Average Calibration Error (AverageCE). AverageCE uses the same fixed-width bins as ECE, but assigns equal weight to each non-empty bin instead of weighting bins by their sample counts. Let $\mathcal{B}^+$ denote the set of non-empty bins. Then,

$$\text{AverageCE} = \frac{1}{|\mathcal{B}^+|}\sum_{b\in\mathcal{B}^+}\left|\text{Pred}(\mathcal{I}_b) - \text{Succ}(\mathcal{I}_b)\right| .$$

Compared with ECE, AverageCE gives the same importance to calibration errors in different prediction ranges regardless of how many samples fall into each range.

C.3 Baselines

BetaPRM. The core of BetaPRM (Li et al., 2026b) lies in modeling the process reward as a Beta distribution, rather than using the point estimation. Specifically, BetaPRM assume a binomial

generative process for the successful continuations: $K_t \mid q_t \sim \text{Binomial}(N, q_t)$. It models q_t with a Beta belief, $q_t \sim \text{Beta}(\alpha_t, \beta_t)$. Then the Beta distribution is reparameterized by its mean $\mu_t = \alpha_t/(\alpha_t + \beta_t)$ and concentration $\kappa_t = \alpha_t + \beta_t$. At the t-th `<prm>` marker, the base model produces a hidden state h_t and vocabulary logits z_t. μ_t is computed as:

$$\mu_t = \frac{\exp(z_t^{\text{yes}})}{\exp(z_t^{\text{yes}}) + \exp(z_t^{\text{no}})},$$

where z_t^{yes} and z_t^{no} denote the logits of two tokens `yes` and `no`. The reliability is represented by the concentration parameter κ_t:

$$\kappa_t = \text{softplus}(g_\phi(h_t)) + \kappa_{\min},$$

where g_ϕ is a lightweight linear head and $\kappa_{\min}$ is a small positive number for numerical stability. Then $\alpha_t = \mu_t\kappa_t$ and $\beta_t = (1 - \mu_t)\kappa_t$.

BetaPRM is trained with the Beta-Binomial loss,

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

where

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

represents the predictive probability of the observed count. BetaPRM also adds a regularization loss to penalize the product of disagreement and concentration:

$$\mathcal{L}_{\text{reg}} = \lambda_{\text{reg}} \frac{1}{|\mathcal{P}|}\sum_{t\in\mathcal{P}} \left|\text{sg}(\mu_t) - \frac{K_t}{N}\right| \kappa_t,$$

where $\text{sg}(\cdot)$ is the stop-gradient operation. The overall training loss is

$$\mathcal{L} = \mathcal{L}_{\text{Beta-Binomial}} + \mathcal{L}_{\text{reg}}.$$

We reproduce BetaPRM following its official implementation[1].

DreamPRM. DreamPRM (Cao et al., 2026) is a domain-reweighted multimodal process reward training framework that addresses the dataset quality imbalance issue in multimodal reasoning. DreamPRM dynamically adjusts the importance of different training domains with bi-level optimization (Talbi, 2013). It solves the following optimization problem:

$$\begin{aligned} \min_{\alpha} \quad & \mathcal{L}_{\text{meta}}\big(\mathcal{D}_{\text{meta}}, \phi^*(\alpha)\big) \\ \text{s.t.} \quad & \phi^*(\alpha) = \arg\min_{\phi} \mathcal{L}_{\text{tr}}(\mathcal{D}_{\text{tr}}, \phi, \alpha) \end{aligned}$$

where $\mathcal{L}_{\text{tr}}(\mathcal{D}_{\text{tr}}, \phi, \alpha) = \sum_{k=1}^{K} \alpha_k \mathcal{L}_{\text{tr}}(\mathcal{D}_k, \phi)$, α_k is the weight coefficient for dataset $\mathcal{D}_k$, $\mathcal{L}_{\text{tr}}(\mathcal{D}_k, \phi)$ is the standard PRM loss with MSE loss on $\mathcal{D}_k$. $\mathcal{D}_{\text{meta}}$ is a validation dataset for testing the generalization of PRMs. $\mathcal{L}_{\text{meta}}$ is the meta loss on $\mathcal{D}_{\text{meta}}$ to penalize the inconsistency of the aggregated PRM score with the ground-truth reward. We implement DreamPRM based on its official codebase[2], and replace its training dataset with VisualPRM400K-v1.1.

QR-PRM. QR-PRM (Park et al., 2026) calibrates the PRM for a target policy through quantile regression on a calibration set and further uses the calibrated reward for instance-adaptive test-time scaling (IAS). Its key observation is that the PRMs often overestimate the process reward due to mismatch of the rollout policy at test time and the policy for collecting the supervision data.

Calibration Set Construction. Given a target policy π, QR-PRM first constructs a policy-specific calibration set. For each validation question q, π generates multiple reasoning trajectories. Each trajectory is decomposed into reasoning prefixes, and for every prefix c_t, the empirical successful rate $\tilde{p}_t$ is computed with Monte-Carlo rollouts generated by π. The resulting tuples $\mathcal{D}_{\text{cal}} = \{(c_t, \tilde{p}_t)\}$ constitute the calibration set used to adapt the PRM to the target policy.

[1] https://github.com/JinYuanLi0012/Beta-Binomial-PRM.git
[2] https://github.com/coder-qicao/DreamPRM.git

Quantile Regression. QR-PRM predicts quantiles of the process reward on $\mathcal{D}_{\text{cal}}$. Specifically, the original PRM linear head is replaced with a quantile prediction head. For a target quantile level τ_j, the model predicts $\hat{r}_t^{(\tau_j)} = \sigma\left(g_{\phi,j}(h_t)\right)$, where h_t denotes the hidden representation of c_t and g_ϕ is the quantile prediction head. The PRM is fine-tuned on $\mathcal{D}_{\text{cal}}$ using the weighted quantile loss

$$\mathcal{L} = \frac{1}{|\mathcal{D}_{\text{cal}}|} \sum_t \frac{1}{J} \sum_{j=1}^{J} \left[\tau_j \max\left(0, \tilde{p}_t - \hat{r}_t^{(\tau_j)}\right) + (1-\tau_j) \max\left(0, \hat{r}_t^{(\tau_j)} - \tilde{p}_t\right) \right].$$

The asymmetric loss allows different quantiles of the success probability to be estimated. The original work predicts the 10%, 50%, and 90% quantiles. Since QR-PRM is not designed as a multimodal PRM, we implement our own multimodal version following its official codebase[3].

Instance-Adaptive Scaling (IAS). QR-PRM further uses the calibrated process reward to dynamically allocate test-time computation. For Best-of-N, given a calibrated reward $\hat{r}^{(\tau)}$ and a target success probability C, the required number of samples is estimated as

$$N_{\text{IAS}} = \min\left(\left\lceil \frac{\log(1-C)}{\log(1-\hat{r}^{(\tau)})} \right\rceil, N_{\max} \right),$$

where $N_{\max}$ denotes the maximum sampling budget. Therefore, more difficult questions receive larger budgets. The original work adopts a conservative lower quantile ($\tau = 0.1$) for IAS.

Self-Consistency. Self-Consistency (Wang et al., 2022) is a verifier-free test-time scaling baseline that improves reasoning by aggregating multiple sampled responses. Given a problem, we sample N candidate reasoning trajectories from the rollout policy and extract their final answers. The answer receiving the majority vote is selected as the final prediction. Unlike PRM-guided Best-of-N, Self-Consistency does not rely on an external reward model and selects the output solely based on agreement among sampled responses.

Temperature Scaling. Temperature scaling (TS) is a parametric post-hoc calibration method that rescales the output logits of a pretrained PRM using a single temperature parameter. Suppose the PRM produces two logits z_t^{yes} and z_t^{no} at the `<prm>` marker. The calibrated process reward is computed as

$$\hat{r}_t^{\text{TS}} = \frac{\exp\left(z_t^{\text{yes}}/T\right)}{\exp\left(z_t^{\text{yes}}/T\right) + \exp\left(z_t^{\text{no}}/T\right)},$$

where $T > 0$ is the temperature parameter. The parameter T is selected on the calibration set by minimizing a calibration objective, such as the Brier score. Temperature scaling preserves the ranking induced by the original logits while adjusting the sharpness of the predicted probabilities.

Isotonic Regression. Isotonic regression (IR) is a non-parametric calibration method that learns a monotonic mapping from the original PRM prediction to the empirical success probability. Given calibration pairs $\{(\hat{r}_t, \tilde{p}_t)\}$, where $\hat{r}_t$ is the original PRM reward and $\tilde{p}_t$ is the corresponding empirical success rate, isotonic regression learns a non-decreasing function $g_{\text{IR}} : [0,1] \to [0,1]$ by solving

$$g_{\text{IR}} = \arg\min_{g \in \mathcal{G}_{\text{mono}}} \sum_t \left(g(\hat{r}_t) - \tilde{p}_t\right)^2,$$

where $\mathcal{G}_{\text{mono}}$ denotes the set of non-decreasing functions. The calibrated reward is then

$$\hat{r}_t^{\text{IR}} = g_{\text{IR}}(\hat{r}_t).$$

In practice, the mapping is fitted using the Pool Adjacent Violators Algorithm (PAVA), resulting in a piecewise-constant monotonic calibration function. Unlike temperature scaling, isotonic regression does not assume a specific parametric form for the calibration mapping.

Histogram Binning. Histogram binning (HB) is another non-parametric post-hoc calibration method. It partitions the prediction interval $[0,1]$ into B equal-width bins $\{\mathcal{B}_b\}_{b=1}^{B}$. For each bin, the calibrated value is defined as the average empirical success rate of the calibration samples whose original PRM predictions fall into that bin:

$$\mu_b = \frac{1}{|\mathcal{I}_b|} \sum_{t \in \mathcal{I}_b} \tilde{p}_t, \qquad \mathcal{I}_b = \{t : \hat{r}_t \in \mathcal{B}_b\}.$$

[3] https://github.com/young-j-park/prm-calibration.git

For a test prefix whose original PRM prediction $\hat{r}_t$ falls into $\mathcal{B}_b$, the calibrated process reward is

$$\hat{r}_t^{\mathrm{HB}} = \mu_b.$$

Thus, histogram binning replaces each raw prediction with the empirical success rate observed in the corresponding confidence interval.

D Implementation Details

In this section, we present the implementation details of BayesianPRM.

Codebase. BayesianPRM is implemented based on BetaPRM codebase[4]. We thank the authors of BetaPRM for releasing such a comprehensive training and evaluation pipeline.

Model Architecture. *(1) Reward Ensemble*. The reward ensemble is built upon the InternVL3-8B backbone and an ensemble of M reward heads $\{g_m\}_{m=1}^{M}$. For each reasoning prefix c_t, we extract the final-layer hidden representation $z_t \in \mathbb{R}^{H_1}$ at the corresponding `<prm>` position. Each reward head is parameterized by a two-layer MLP with one hidden layer with dimension H_2. Instead of instantiating M separate modules, we stack the parameters of all ensemble members along an additional ensemble dimension, resulting in an ensemble architecture of $[M, H_1, H_2, 1]$. We then apply a sigmoid activation to constrain the reward values to the range $[0, 1]$. *(2) Randomized Prior Network*. To improve the diversity of the reward ensemble, we associate the reward heads with an randomized prior network $\{f_m\}_{m=1}^{M}$. The prior network adopts the same MLP architecture of $[M, H_1, H_2, 1]$ as the reward heads, while its parameters are randomly initialized and kept fixed. For the m-th ensemble member, the reward is given by $r_m(c_t) = \sigma(g_m(z_t) + \lambda_{\text{prior}} f_m(z_t))$. That is, the reward logit is jointly determined by the learned reward head and the corresponding prior network. Since different reward heads are equipped with distinct fixed prior functions, each member learns a residual correction on top of a different functional initialization. This discourages the ensemble members from collapsing to identical reward functions under the same supervision and thereby improves the diversity of the reward ensemble. *(3) Belief Network*. The belief network is designed to estimate a prefix-dependent reliability distribution over the M reward hypotheses. For each reasoning prefix c_t, we concatenate its hidden representation $z_t \in \mathbb{R}^{H_1}$ with the reward predictions of all ensemble members, $\boldsymbol{r}(c_t) = [r_1(c_t), \ldots, r_M(c_t)]^\top$, as the input to the belief network: $x_t = [z_t; \boldsymbol{r}(c_t)] \in \mathbb{R}^{H_1+M}$. The belief network $w_\nu(\cdot)$ is parameterized by a two-layer MLP with one hidden layer of dimension H_2, resulting in an architecture of $[H_1+M, H_2, M]$. Specifically, the input x_t is forwarded through w_v, which outputs M belief logits. We then apply a softmax function over the ensemble dimension to obtain the reliability-aware belief $\boldsymbol{\alpha}^{\text{rel}}(c_t) = \mathrm{Softmax}(w_\nu(x_t))$. During belief-network training, the backbone, reward heads, and prior networks are kept frozen, and only the parameters of the belief network are optimized.

Training Procedure. The training procedure of BayesianPRM consists of two stages: reward ensemble training and belief network training. *(1) Reward Ensemble Training.* In this stage, we freeze the vision encoder while fine-tuning the language model and all reward heads with $\mathcal{D}_{\text{train}}$. Each reward head is optimized using the standard cross-entropy loss:

$$\mathcal{L}_{\mathrm{CE}} = -\frac{1}{M|\mathcal{D}_{\text{train}}|} \sum_{c_t \in \mathcal{D}_{\text{train}}} \sum_{m=1}^{M} \left[\hat{q}_t \log r_m(c_t) + (1-\hat{q}_t)\log(1-r_m(c_t))\right], \tag{16}$$

where $\hat{q}_t$ is the Monte-Carlo label, $r_m(c_t)$ is the prediction of m-th reward head. *(2) Belief Network Training*. During belief network training, we freeze the entire backbone and all reward heads, and update only the belief network w_ν on $\mathcal{D}_{\text{val}}$ using the ELBO objective in Equation 6. After training, given a reasoning prefix c_t, we obtain the original ensemble reward predictions $\boldsymbol{r}(c_t)$ together with the reliability-aware belief $\boldsymbol{\alpha}^{\text{rel}}(c_t)$.

Inference Procedure. During inference, we first calibrate the reliability-aware belief with the conservatism-aware adjustment to obtain $\boldsymbol{\alpha}^{\text{hyb}}(c_t)$, as defined in Equation 9. We then compute the final hybrid reward by taking a weighted sum of the reward ensemble predictions using $\boldsymbol{\alpha}^{\text{hyb}}(c_t)$: $\tilde{r}(c_t) = [\boldsymbol{\alpha}^{\text{hyb}}(c_t)]^\top \boldsymbol{r}(c_t)$. For test-time scaling, we need to assign a single reward to each complete

[4] https://github.com/JinYuanLi0012/Beta-Binomial-PRM.git

Table 6: Hyperparameters for BayesianPRM.

Item	Value
Backbone	InternVL3-8B
Optimizer	AdamW (Loshchilov & Hutter, 2017)
Learning Rate	1×10^{-5}
Weight Decay	0.05
Learning Rate Scheduler	Cosine
Warmup Ratio	0.05
Epochs	1
Global Batch Size	512
Ensemble / Belief Data Split	80% / 20%
Max Sequence Length	8192
Ensemble Size M	20
Backbone Hidden Dimension H_1	3584
Reward Head / Belief Network Hidden Dimension H_2	256
Reliability Coefficient β_1	0.05
Conservatism Coefficient β_2	0.05
Prior Weighting Coefficient λ_{prior}	10

reasoning trajectory. Given a trajectory τ consisting of T reasoning steps, we aggregate the process rewards by averaging them across all steps: $R(\tau) = \frac{1}{T}\sum_{t=1}^{T} \tilde{r}(c_t)$.

We present the pseudo-code of BayesianPRM in Algorithm 1.

Algorithm 1 BayesianPRM

Require: Training dataset $\mathcal{D}_{\text{train}} = \{c_t, K_t, N_t\}_{t=1}^{T}$, validation dataset $\mathcal{D}_{\text{val}} = \{c_t, K_t, N_t\}_{t=1}^{T'}$, LLM backbone
Initialization: ensemble size M, reward ensemble $\{r_{\theta_m}\}_{m=1}^{M}$, randomized prior network $\{f_{\phi_m}\}_{m=1}^{M}$, belief network w_ν, reliability coefficient β_1, conservatism coefficient β_2, prior weighting coefficient λ_{prior}
// Train the Reward Ensemble
for each gradient step **do**
  Compute loss $\mathcal{L}_{\text{CE}} = -\mathbb{E}_{c_t \in \mathcal{D}_{\text{train}}} \frac{1}{M} \sum_{m=1}^{M} [\hat{q}_t \log r_m(c_t) + (1-\hat{q}_t)\log(1-r_m(c_t))]$
  Update $\{r_{\theta_m}\}_{m=1}^{M}$ using $\mathcal{L}_{\text{CE}}$
end for
// Train the Belief Network
for each gradient step **do**
  Compute loss $\mathcal{L}_{\text{ELBO}} = -\mathbb{E}_{c_t \in \mathcal{D}_{\text{val}}} \sum_{m=1}^{M} \alpha_m^{\text{rel}}(c_t)(K_t \log r_m(c_t) + (N_t - K_t)\log(1-r_m(c_t)) - \beta_1 \log(M\alpha_m^{\text{rel}}(c_t)))$
  Update w_ν using $\mathcal{L}_{\text{ELBO}}$
end for
Obtain the reliability-aware belief $\boldsymbol{\alpha}^{\text{rel}}(c_t)$
// Inference
Conservatively calibrate the reliability-aware belief, and obtain the hybrid belief $\alpha_m^{\text{hyb}}(c_t) = \frac{\alpha_m^{\text{rel}}(c_t)\exp(-r_m(c_t)/\beta_2)}{\sum_{j=1}^{M}\alpha_j^{\text{rel}}(c_t)\exp(-r_j(c_t)/\beta_2)}, \quad m = 1, 2, ..., M$
Obtain the hybrid reward by taking a weighted sum of the reward ensemble predictions with $\boldsymbol{\alpha}^{\text{hyb}}(c_t)$: $\tilde{r}(c_t) = [\boldsymbol{\alpha}^{\text{hyb}}(c_t)]^\top \boldsymbol{r}(c_t)$

Hyperparameter Setup. We list the main hyperparameters of BayesianPRM in Table 6.

Table 7: **PRM accuracy evaluation results for Qwen3VL-8B.** We evaluate Brier Score, Positive Brier Score, ECE, and AverageCE on four benchmarks: MathVision, the Vision Only subset of MathVerse (denoted as MathVerse-VO), MathVista, and MMStar. ↓ indicates that lower values are better. ± denotes the standard deviation across three random training seeds.

	MathVision	MathVerse-VO	MathVista	MMStar	Average
			Brier Score		
Standard PRM	0.2540±0.0128	0.2695±0.0210	0.2223±0.0143	0.2506±0.0088	0.2491
BetaPRM	0.2854±0.0259	0.2901±0.0220	0.2267±0.0166	0.2540±0.0192	0.2641
DreamPRM	0.2869±0.0234	0.3221±0.0202	0.2363±0.0051	0.2588±0.0120	0.2760
BayesianPRM	**0.2002**±0.0187	**0.2439**±0.0261	**0.2118**±0.0242	**0.2262**±0.0105	**0.2205**
			Positive Brier Score		
Standard PRM	0.2163±0.0152	0.2587±0.0118	0.1743±0.0244	0.2393±0.0130	0.2222
BetaPRM	0.2569±0.0226	0.2413±0.0174	0.1890±0.0125	0.2448±0.0096	0.2330
DreamPRM	0.2607±0.0287	0.3054±0.0190	0.1966±0.0118	0.2307±0.0162	0.2484
BayesianPRM	**0.1393**±0.0130	**0.1747**±0.0231	**0.1410**±0.0276	**0.2049**±0.0133	**0.1650**
			ECE		
Standard PRM	0.2809±0.0236	0.2231±0.0204	0.1628±0.0181	0.2364±0.0042	0.2258
BetaPRM	0.3371±0.0177	0.2590±0.0262	0.1689±0.0213	0.2368±0.0160	0.2505
DreamPRM	0.3257±0.0216	0.2588±0.0149	0.1842±0.0106	0.2219±0.0165	0.2477
BayesianPRM	**0.1763**±0.0115	**0.1485**±0.0231	**0.1181**±0.0144	**0.1898**±0.0139	**0.1582**
			AverageCE		
Standard PRM	0.2746±0.0227	0.1857±0.0128	0.1421±0.0147	0.2076±0.0122	0.2025
BetaPRM	0.2676±0.0164	0.2242±0.0142	0.1715±0.0233	0.1600±0.0169	0.2058
DreamPRM	0.2856±0.0221	0.2113±0.0157	0.1742±0.0217	0.1820±0.0070	0.2133
BayesianPRM	**0.2047**±0.0121	**0.1405**±0.0229	**0.1064**±0.0063	**0.1581**±0.0147	**0.1524**

E Additional Experimental Results

In this section, we provide additional experimental results and analyses omitted from the main text due to space limits, offering a more comprehensive evaluation of BayesianPRM.

E.1 Results with Qwen3VL-8B

In Section 5.1 and Section 5.2, we use InternVL3-8B as the test-time rollout policy to evaluate PRM accuracy and test-time scaling performance. In this section, we replace it with Qwen3-VL-8B-Instruct (Team, 2025) to examine the generalizability of BayesianPRM across different rollout backbones.

E.1.1 Process Reward Accuracy Evaluation

Datasets. The training dataset is kept consistent with Section 5.1, using VisualPRM400K-v1.1. For BayesianPRM, we randomly use 80% of the data to train the reward ensemble and the remaining 20% to train the belief network. For the evaluation dataset, we still use MathVision, MathVerse-VO, MathVista and MMStar. For each image-question pair, we first use Qwen3VL-8B-Instruct to generate a reasoning response, from which intermediate reasoning prefixes are extracted. Starting from each prefix c_t, we perform 16 Monte-Carlo rollouts using Qwen3VL-8B-Instruct. Each rollout is then evaluated by Qwen2.5-32B-Instruct as the judge model to determine its correctness. The proportion of correct rollouts is treated as the empirical success rate $\hat{p}_t$. The $(c_t, \hat{p}_t)$ pairs constitute $\mathcal{D}_{\text{test}}$ for evaluation.

Evaluation Metrics. Following Section 5.1, we use Brier Score, Positive Brier Score, Expected Calibration Error (ECE) and Average Calibration Error (AverageCE) as the evaluation metrics.

Baselines. Consistent with Section 5.1, we choose Standard PRM, BetaPRM, and DreamPRM as baselines.

Table 8: **PRM-guided test-time scaling performance comparison.** We compare Best-of-N ($N = 16$) and Instance-Adaptive Scaling (IAS, $N_{\max} = 16$) on MathVision, MathVerse-VO, MathVista and MMStar. $\uparrow$ indicates that higher values are better. $\pm$ denotes the standard deviation across three random training seeds.

	MathVision	MathVerse-VO	MathVista	MMStar	Average
Qwen3VL-8B	20.07	50.51	70.30	56.67	49.39
Best-of-N ($N = 16$)					
Self-Consistency	**22.04**	54.19	73.20	59.33	52.19
Standard PRM	20.86±0.36	53.52±0.64	74.97±1.04	59.61±0.27	52.24
BetaPRM	21.14±0.57	53.39±0.33	74.79±0.70	60.07±0.44	52.35
DreamPRM	21.06±0.62	53.71±0.20	74.15±1.22	60.27±0.65	52.30
BayesianPRM	21.52±0.39	**54.29**±0.56	**75.71**±0.78	**60.69**±0.53	**53.05**
IAS ($N_{\max} = 16$)					
Standard PRM	20.73±0.46	52.73±0.49	73.81±0.81	59.40±0.32	51.67
BetaPRM	21.08±0.71	52.63±0.30	73.86±0.68	59.35±0.43	51.73
DreamPRM	20.89±1.36	53.11±0.48	74.06±0.74	59.28±0.56	51.84
BayesianPRM	**21.46**±0.72	**54.07**±0.45	**74.65**±0.77	**60.20**±0.48	**52.60**

Results. As shown in Table 7, BayesianPRM consistently achieves the best PRM accuracy across all four benchmarks and all evaluation metrics when Qwen3-VL-8B-Instruct is used as the rollout policy. Compared with Standard PRM, BayesianPRM reduces the average Brier Score from 0.2491 to 0.2205 and the Positive Brier Score from 0.2222 to 0.1650. More substantial improvements are observed in calibration metrics, where the average ECE decreases from 0.2258 to 0.1582 and AverageCE from 0.2025 to 0.1524. BayesianPRM also consistently outperforms BetaPRM and DreamPRM across all metrics. These results further demonstrate that the improved reward estimation and calibration of BayesianPRM generalize to different rollout backbones.

E.1.2 Test-Time Scaling Performance

Experimental Setup. The training dataset $\mathcal{D}_{\text{train}}$ and evaluation datasets $\mathcal{D}_{\text{test}}$ are kept consistent with those used in Section E.1.1. We compare the performance of Best-of-N and IAS. For Best-of-N, we set $N = 16$. For each $c_t \in \mathcal{D}_{\text{test}}$, Qwen3VL-8B-Instruct generates 16 candidate responses, which are scored by different PRMs, and the response with the highest score is selected. For IAS, we set $N_{\max} = 16$ and dynamically allocate the rollout budget; We use Standard PRM, BetaPRM, and DreamPRM as baselines. For Best-of-N , we add an additional Self-Consistency baseline.

Results. As shown in Table 8, BayesianPRM consistently improves PRM-guided test-time scaling with Qwen3-VL-8B-Instruct as the rollout policy. Under Best-of-N with $N = 16$, BayesianPRM achieves the highest average accuracy of 53.05, outperforming Standard PRM and the second-best PRM baseline, BetaPRM, by 0.81 and 0.70 points, respectively. It achieves the best performance on MathVerse-VO, MathVista, and MMStar, while ranking second on MathVision. Under IAS with $N_{\max} = 16$, BayesianPRM achieves the best performance on all four benchmarks and the highest average accuracy of 52.60, exceeding Standard PRM and the second-best method, DreamPRM, by 0.93 and 0.76 points, respectively. These results show that the test-time scaling improvements of BayesianPRM remain consistent when using different rollout backbones.

E.2 Evaluation on VisualProcessBench

In this part, we further evaluate BayesianPRM on VisualProcessBench (Wang et al., 2026), a benchmark designed to evaluate the ability of PRMs to identify erroneous steps in multimodal reasoning. It contains 2,866 samples and 26,950 human-annotated step-level correctness labels, covering five source benchmarks: MMMU (Yue et al., 2024), MathVision, MathVerse, DynaMath (Zou et al., 2025), and WeMath (Qiao et al., 2025). Each reasoning step is annotated as correct, incorrect, or neutral. Following prior work, we exclude neutral steps from evaluation. We report macro-F1

Table 9: **Evaluation results on VisualProcessBench.** Each source column reports macro-F1 on the corresponding subset, while Overall denotes the micro-F1 computed over all non-neutral annotated steps. ± denotes the standard deviation across three random training seeds.

Model	Overall	MathVision	MathVerse	MMMU	DynaMath	WeMath
Standard PRM	59.79 ±0.41	57.03±0.16	59.65±0.26	62.23±0.42	59.71±0.24	63.13±0.60
BetaPRM	60.83±0.23	57.83±0.38	60.40±0.44	60.59±0.26	**61.33**±0.62	63.47±0.48
DreamPRM	60.69±0.32	58.13±0.47	59.44±0.29	61.22±0.54	61.06±0.15	63.59±0.27
BayesianPRM	**62.23**±0.29	**61.58**±0.54	**61.31**±0.31	**63.39**±0.36	61.12±0.47	**64.06**±0.22

for each source benchmark and compute the overall score as micro-F1 over all non-neutral annotated steps pooled across sources. Standard PRM, BetaPRM, DreamPRM, and BayesianPRM are evaluated on VisualProcessBench using models trained on VisualPRM400K-v1.1. This evaluation directly measures the step-level error detection ability of different PRMs, providing a complementary evaluation to the calibration and test-time scaling results. We use a fixed global classification threshold of 0.8 for all PRMs. Steps with scores above 0.8 are classified as correct, and the same threshold is applied uniformly across all five source benchmarks.

Results. As shown in Table 9, BayesianPRM achieves the highest overall score of 62.23 on VisualProcessBench, outperforming Standard PRM and the second-best method, BetaPRM, by 2.44 and 1.40 points, respectively. BayesianPRM achieves the best performance on four of the five subsets, with particularly clear improvements on MathVision, where it exceeds the second-best method by 3.45 points. On DynaMath, BayesianPRM is only 0.21 points lower than BetaPRM. These results show that BayesianPRM provides more accurate step-level reward predictions across diverse multimodal reasoning tasks.

E.3 Evaluation with Calibration Sets

In this section, we compare BayesianPRM with additional baselines, including QR-PRM and post-hoc calibration methods including Temperature Scaling, Isotonic Regression, and Histogram Binning, as described in Appendix C.3. Since these methods require additional calibration sets, we exclude them from the main-text comparison to ensure a fair evaluation. We also exclude OT-PRM from the baselines because its official implementation has not been publicly released.

Experimental Setup. We randomly split VisualPRM400K-v1.1 into an 80% training set and a disjoint 20% calibration set. The Monte-Carlo labels for the calibration set are obtained using InternVL3-8B with 16 rollouts per sample. For BayesianPRM, we train the reward ensemble on the 80% training split and train the belief network on the held-out 20% calibration split. For QR-PRM, we initialize from a Standard PRM trained only on the same 80% training split and fine-tune it on the held-out calibration split following the procedure described in Appendix C.3. For Temperature Scaling, Isotonic Regression, and Histogram Binning, we use the same 80%-trained Standard PRM as the backbone and fit the calibration mappings exclusively on the held-out calibration split. All PRMs use InternVL3-8B as the LLM backbone. Following the setup in Section 5, we evaluate these methods in terms of both process reward accuracy and test-time scaling performance. The 80%-trained Standard PRM is used only for the calibration comparison in this section, while the Standard PRM reported in the main experiments is trained on the full VisualPRM400K-v1.1 dataset.

Process Reward Accuracy Evaluation. As shown in Table 10, BayesianPRM achieves the best results on every benchmark under all four evaluation metrics. Compared with QR-PRM, BayesianPRM reduces the average Brier Score from 0.1668 to 0.1231 and the Positive Brier Score from 0.1508 to 0.1095. Similar improvements are observed in calibration metrics, with the ECE decreasing from 0.1869 to 0.1264 and AverageCE from 0.1728 to 0.1267. In contrast, the three post-hoc calibration methods, including Temperature Scaling, Isotonic Regression, and Histogram Binning, show limited effectiveness in this setting and consistently underperform QR-PRM and BayesianPRM across most metrics. These results show that BayesianPRM remains effective when compared with methods that explicitly leverage an additional calibration set.

Test-Time Scaling Performance. As shown in Table 11, BayesianPRM achieves the best overall test-time scaling performance under both fixed-budget Best-of-N and instance-adaptive scaling.

Table 10: **PRM accuracy evaluation results with calibration sets.** We evaluate Brier Score, Positive Brier Score, ECE, and AverageCE on four benchmarks: MathVision, the Vision Only subset of MathVerse (denoted as MathVerse-VO), MathVista, and MMStar. TS, IR, HB denote Temperature Scaling, Isotonic Regression, Histogram Binning, respectively. ↓ indicates that lower values are better. ± denotes the standard deviation across three random training seeds.

	MathVision	MathVerse-VO	MathVista	MMStar	Average
			Brier Score ↓		
QR-PRM	0.1634±0.0166	0.1899±0.0147	0.1343±0.0124	0.1795±0.0262	0.1668
TS	0.2931±0.0216	0.2652±0.0124	0.2145±0.0189	0.2667±0.0160	0.2599
IR	0.2568±0.0357	0.2420±0.0324	0.2001±0.0164	0.2504±0.0293	0.2373
HB	0.3146±0.0250	0.2847±0.0127	0.2394±0.0165	0.2781±0.0277	0.2792
BayesianPRM	**0.1035**±0.0142	**0.1422**±0.0169	**0.0946**±0.0125	**0.1521**±0.0178	**0.1231**
			Positive Brier Score ↓		
QR-PRM	0.1548±0.0247	0.1697±0.0206	0.1182±0.0122	0.1603±0.0281	0.1508
TS	0.2855±0.0157	0.2437±0.0231	0.1964±0.0152	0.2503±0.0204	0.2440
IR	0.2416±0.0214	0.2309±0.0195	0.1874±0.0267	0.2377±0.0347	0.2244
HB	0.2988±0.0273	0.2713±0.0215	0.2106±0.0155	0.2622±0.0303	0.2607
BayesianPRM	**0.0914**±0.0065	**0.1337**±0.0143	**0.0735**±0.0157	**0.1394**±0.0166	**0.1095**
			ECE ↓		
QR-PRM	0.2083±0.0255	0.2105±0.0136	0.1442±0.0246	0.1846±0.0309	0.1869
TS	0.3357±0.0260	0.3102±0.0391	0.2451±0.0241	0.2683±0.0272	0.2898
IR	0.2943±0.0323	0.3127±0.0224	0.2670±0.0149	0.2463±0.0266	0.2801
HB	0.3427±0.0266	0.2896±0.0199	0.2613±0.0163	0.2951±0.0295	0.2972
BayesianPRM	**0.1602**±0.0131	**0.1578**±0.0182	**0.0528**±0.0095	**0.1349**±0.0163	**0.1264**
			AverageCE ↓		
QR-PRM	0.2088±0.0249	0.1867±0.0280	0.1261±0.0156	0.1694±0.0142	0.1728
TS	0.3021±0.0141	0.2752±0.0256	0.2161±0.0340	0.2430±0.0244	0.2591
IR	0.2710±0.0130	0.2899±0.0291	0.2351±0.0201	0.2244±0.0107	0.2551
HB	0.3281±0.0142	0.2640±0.0190	0.2388±0.0203	0.2542±0.0082	0.2713
BayesianPRM	**0.1720**±0.0135	**0.1453**±0.0146	**0.0792**±0.0042	**0.1103**±0.0060	**0.1267**

With $N = 16$, BayesianPRM obtains the highest average accuracy of 56.75, outperforming QR-PRM by 0.74 points, and achieves the best performance on all four benchmarks. Under IAS with $N_{\max} = 16$, BayesianPRM again achieves the highest average accuracy of 56.14, exceeding QR-PRM by 0.64 points. It performs best on three of the four benchmarks and is only slightly lower than QR-PRM on MMStar. In contrast, the post-hoc calibration methods, including Temperature Scaling, Isotonic Regression, and Histogram Binning, show relatively weaker test-time scaling performance. These results further demonstrate the effectiveness of BayesianPRM even when compared with methods that leverage an additional calibration set.

E.4 Extended Ablation Study

In this section, we provide additional ablation study results omitted from the main text due to space limits.

Comparison with Vanilla Ensemble PRM. We construct a vanilla ensemble PRM by uniformly averaging the predictions of all ensemble members, without performing belief updating or calibration. We compare its test-time scaling performance with BayesianPRM, with the results reported in Table 12. It is clear that BayesianPRM consistently outperforms vanilla ensemble PRM, demonstrating the importance of reliability-aware belief updating and conservatism-aware belief calibration.

Effect of β_1. β_1 controls the strength of reliability-aware belief updating, with a smaller β_1 inducing a stronger update. If β_1 is too large, the belief distribution remains overly close to the uniform prior. In contrast, an overly small β_1 produces an excessively sharp belief distribution, which may overfit to label noise. We vary β_1 over $\{0.001, 0.01, 0.05, 0.1\}$ and report the test-time scaling performance

Table 11: **PRM-guided test-time scaling performance comparison with calibration sets.** We compare Best-of-N ($N = 16$) and Instance-Adaptive Scaling (IAS, $N_{\max} = 16$) on MathVision, MathVerse-VO, MathVista and MMStar. $\uparrow$ indicates that higher values are better. $\pm$ denotes the standard deviation across three random training seeds.

	MathVision	MathVerse-VO	MathVista	MMStar	Average
Best-of-N ($N = 16$) $\uparrow$					
QR-PRM	34.79±0.67	51.26±0.84	69.28±0.73	68.71±0.40	56.01
TS	32.10±1.18	49.66±0.64	69.12±0.48	67.01±0.39	54.47
IR	32.67±1.14	51.26±0.79	68.83±0.85	67.58±0.30	55.09
HB	32.00±1.01	49.13±0.57	69.06±0.82	67.24±1.13	54.36
BayesianPRM	**35.80**±0.56	**51.93**±0.77	**69.56**±0.33	**69.69**±0.90	**56.75**
IAS ($N_{\max} = 16$) $\uparrow$					
QR-PRM	34.25±0.83	50.86±0.45	68.24±0.36	**68.66**±0.69	55.50
TS	32.01±1.20	49.12±0.93	68.74±0.65	67.35±0.46	54.31
IR	32.33±0.96	51.19±0.42	68.41±0.79	67.20±0.63	54.78
HB	31.84±0.35	49.06±1.33	68.51±0.58	67.19±0.75	54.15
BayesianPRM	**35.33**±0.46	**51.61**±0.74	**69.34**±0.71	68.29±0.34	**56.14**

Table 12: **Comparison with Vanilla Ensemble PRM**. We compare Best-of-N ($N = 16$) on MathVision, MathVerse-VO, MathVista and MMStar. $\pm$ denotes the standard deviation across three random training seeds.

	MathVision	MathVerse-VO	MathVista	MMStar	Average
Vanilla Ensemble PRM	34.88±0.47	50.21±0.92	68.26±0.59	68.03±0.88	55.35
BayesianPRM	**35.86**±0.70	**51.40**±1.11	**69.40**±0.53	**69.33**±0.60	**56.50**

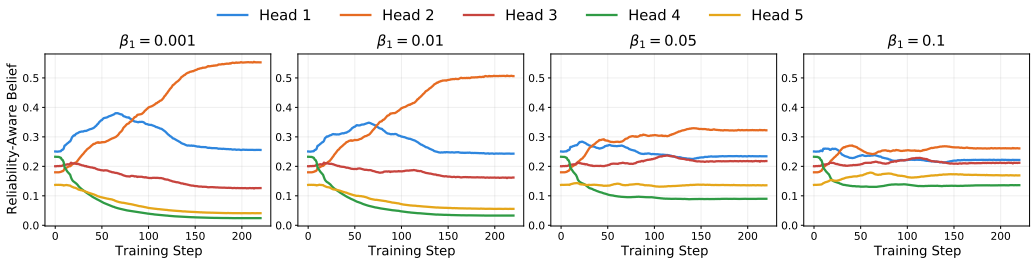


Figure 4: Reliability-aware belief distribution during training. We use $M = 5$ for visualization clarity.

in Table 13. As shown, the performance drops when β_1 is set to either 0.001 or 0.1. We also plot in Figure 4 how the reliability-aware belief over reward heads evolves during training for different values of β_1, with the reward ensemble size fixed to 5. When $\beta_1 = 0.001$ or 0.01, Head 2 tends to dominate while the weights of Head 4 and Head 5 approach zero, indicating an overly concentrated belief distribution. In contrast, when $\beta_1 = 0.1$, the belief remains too evenly distributed across the heads. Therefore, we set $\beta_1 = 0.05$ in our experiments.

Effect of β_2. β_2 controls the strength of conservatism-aware belief calibration, with a smaller β_2 inducing stronger conservative reweighting. If β_2 is too large, reward overestimation may not be suppressed, whereas an overly small β_2 may instead lead to reward underestimation. We vary β_2 over $\{0.001, 0.01, 0.05, 0.1\}$, and Table 14 reports the test-time scaling performance comparison. As shown, performance degrades when β_2 is either too large or too small. We also observe that larger β_2 values are more favorable on MMStar, suggesting that the degree of reward overestimation varies across datasets. Therefore, a single β_2 may not be optimal for all benchmarks. In our experiments,

Table 13: **Ablation Study on** β_1. We compare Best-of-N ($N = 16$) on MathVision, MathVerse-VO, MathVista and MMStar. $\pm$ denotes the standard deviation across three random training seeds.

	MathVision	MathVerse-VO	MathVista	MMStar	Average
$\beta_1 = 0.001$	34.28±0.46	50.30±0.88	68.33±0.67	68.56±0.83	55.37
$\beta_1 = 0.01$	35.67±0.78	51.86±1.27	69.32±0.42	69.07±0.57	56.48
$\beta_1 = 0.05$	35.86±0.70	51.40±1.11	69.40±0.53	69.33±0.60	56.50
$\beta_1 = 0.1$	35.21±0.40	50.35±0.63	68.82±0.27	68.19±0.64	55.64

Table 14: **Ablation Study on** β_2. We compare Best-of-N ($N = 16$) on MathVision, MathVerse-VO, MathVista and MMStar. $\pm$ denotes the standard deviation across three random training seeds.

	MathVision	MathVerse-VO	MathVista	MMStar	Average
$\beta_2 = 0.001$	35.24±0.62	50.62±0.94	68.43±0.46	69.10±0.82	55.85
$\beta_2 = 0.01$	35.78±0.76	50.54±0.85	69.21±0.50	69.39±0.74	56.23
$\beta_2 = 0.05$	35.86±0.70	51.40±1.11	69.40±0.53	69.33±0.60	56.50
$\beta_2 = 0.1$	34.72±0.49	50.26±1.26	69.27±0.61	69.98±0.85	56.06

Table 15: **Ablation Study on** M. We compare Best-of-N ($N = 16$) on MathVision, MathVerse-VO, MathVista and MMStar. $\pm$ denotes the standard deviation across three random training seeds.

	MathVision	MathVerse-VO	MathVista	MMStar	Average
$M = 10$	35.30±0.72	51.42±0.79	68.97±0.42	68.49±0.59	56.05
$M = 20$	35.86±0.70	51.40±1.11	69.40±0.53	69.33±0.60	56.50
$M = 50$	35.89±0.65	51.35±1.30	69.52±0.49	68.99±0.64	56.44
$M = 100$	35.87±0.58	51.46±0.83	69.34±0.60	69.62±0.39	56.57

Table 16: **Ablation Study on** λ_{prior}. We compare Best-of-N ($N = 16$) on MathVision, MathVerse-VO, MathVista and MMStar. $\pm$ denotes the standard deviation across three random training seeds.

	MathVision	MathVerse-VO	MathVista	MMStar	Average
$\lambda_{\text{prior}} = 0.1$	34.93±0.49	49.84±0.53	68.20±0.73	67.77±0.43	55.19
$\lambda_{\text{prior}} = 1$	35.39±0.79	50.27±0.88	68.85±0.56	69.12±0.93	55.91
$\lambda_{\text{prior}} = 10$	35.86±0.70	51.40±1.11	69.40±0.53	69.33±0.60	56.50
$\lambda_{\text{prior}} = 50$	35.41±1.09	51.02±0.69	68.82±0.64	69.24±0.48	56.12

we uniformly set $\beta_2 = 0.05$ unless otherwise specified, rather than extensively tuning it for each benchmark.

Effect of M**.** The ensemble size M determines the number of reward heads. When M is too small, the reward hypothesis space may not be sufficiently covered, whereas an excessively large M may introduce redundant reward heads. We vary M over $\{10, 20, 50, 100\}$, and report the test-time scaling performance in Table 15. We observe that the performance starts to fluctuate when $M > 20$. Therefore, we set $M = 20$ in our experiments.

Effect of λ_{prior}**.** λ_{prior} controls the contribution of the random prior network to each reward head. When λ_{prior} is too small, the reward heads lack sufficient diversity. However, an overly large λ_{prior} may cause the random prior to dominate the learned reward prediction and degrade reward accuracy. We vary λ_{prior} over $\{0.1, 1, 10, 50\}$. As shown in Table 16, the test-time scaling performance degrades when λ_{prior} is either too small or too large. Therefore, we set $\lambda_{\text{prior}} = 10$ throughout our experiments.

F Computational Overhead

BayesianPRM introduces only a small parameter overhead over the standard PRM. The standard PRM contains 7.944B parameters, while BayesianPRM contains 7.982B parameters, corresponding to a 0.48% parameter overhead.

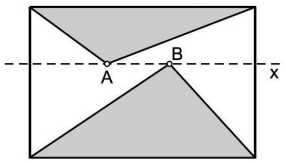
We further compare the training cost on 4 $\times$ H100 GPUs. Training the Standard PRM for one epoch takes approximately 24 hours, whereas the complete training pipeline of BayesianPRM requires less than 20 hours. This difference mainly arises because BayesianPRM trains the reward ensemble on 80% of the data, while the remaining 20% is used to train the lightweight belief network with the backbone frozen. Therefore, in our training setup, BayesianPRM requires less wall-clock training time than Standard PRM.

G Case Study

In this section, we provide a case study of the step-wise rewards assigned by BayesianPRM, as shown in Figure 5. Figure 5 (a) presents a correct reasoning trajectory, where BayesianPRM assigns consistently high rewards to all reasoning steps. Figure 5 (b) presents an incorrect trajectory. At the first erroneous step (Step 8), the reward exhibits a sharp drop from 0.977 to 0.041. These examples illustrate that BayesianPRM can assign high rewards to valid reasoning while identifying erroneous intermediate steps with substantially lower rewards.

H Limitations

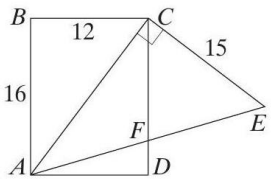
One limitation of our method is that several important hyperparameters, such as β_2, need to be manually specified. However, the degree of reward overestimation can vary across datasets and test-time rollout policies, leading to different optimal hyperparameter values. Moreover, the current conservatism-aware belief calibration performs a one-sided correction toward lower rewards and is therefore most effective when the prediction error is dominated by overestimation. Developing adaptive methods that automatically determine both the strength and direction of calibration would be an interesting direction for future work.



Question: The sum of the areas of the two shaded triangles is $10\,\mathrm{cm}^2$. What is the area of the rectangle?
Choices: (A) 18 (B) 20 (C) 22 (D) 24
Ground Truth: B
Benchmark: MathVision

Step 1: Let the rectangle have width W and height H. **[BayesianPRM: 0.895]**
Step 2: Let h_1 and h_2 be the distances from line x to the top and bottom sides, so $h_1 + h_2 = H$. **[BayesianPRM: 0.906]**
Step 3: The upper triangle has area $\frac{1}{2}Wh_1$. **[BayesianPRM: 0.944]**
Step 4: The lower triangle has area $\frac{1}{2}Wh_2$. **[BayesianPRM: 0.968]**
Step 5: Their total area is $\frac{1}{2}W(h_1 + h_2) = \frac{1}{2}WH$. **[BayesianPRM: 0.968]**
Step 6: Since the shaded area is 10, $\frac{1}{2}WH = 10$. **[BayesianPRM: 0.958]**
Step 7: Therefore, $WH = 20$. **[BayesianPRM: 0.992]**
Step 8: The rectangle has area $20\,\mathrm{cm}^2$. **[BayesianPRM: 0.986]**
Final Answer: B. $20\,\mathrm{cm}^2$ ✓

(a) A correct case.



Question: In rectangle $ABCD$, $AB = 16$, $BC = 12$, $AC \perp CE$, and $CE = 15$. If $F = AE \cap CD$, find the area of triangle ACF.
Ground Truth: 75
Benchmark: MathVision

Step 1: Identify the geometric setting: rectangle $ABCD$ with $AB = 16$, $BC = 12$, $AC \perp CE$, $CE = 15$, and $F = AE \cap CD$. **[BayesianPRM: 0.785]**
Step 2: Compute the diagonal: $AC = \sqrt{16^2 + 12^2} = 20$. **[BayesianPRM: 0.865]**
Step 3: Since $AC \perp CE$ and $CE = 15$, triangle ACE is a right triangle. **[BayesianPRM: 0.922]**
Step 4: Set coordinates $A = (0, 0)$, $B = (0, 16)$, $C = (12, 16)$, and $D = (12, 0)$. **[BayesianPRM: 0.943]**
Step 5: Because the slope of AC is $\frac{4}{3}$, the slope of CE is $-\frac{3}{4}$. Using the perpendicular condition and $CE = 15$, solve for E. **[BayesianPRM: 0.960]**
Step 6: This gives two candidates, and the one consistent with the figure is $E = (24, 7)$. **[BayesianPRM: 0.913]**
Step 7: Hence line AE has equation $y = \frac{7}{24}x$. **[BayesianPRM: 0.977]**
Step 8: Since CD is the vertical line $x = 12$, substitute into $y = \frac{7}{24}x$ and obtain $y = \frac{7}{24} \times 12 = 7$. Thus, $F = (12, 7)$. **[BayesianPRM: 0.041]**
Step 9: Using $A = (0, 0)$, $C = (12, 16)$, and $F = (12, 7)$, compute the area of triangle ACF to be 54. **[BayesianPRM: 0.017]**
Final Answer: 54 ×

(b) An incorrect case.

Figure 5: A case study of BayesianPRM's step-wise evaluation.