# OPTIMAL AGGREGATION OF LLM AND PRM SIGNALS FOR EFFICIENT TEST-TIME SCALING

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

## ABSTRACT

Process reward models (PRMs) are a cornerstone of test-time scaling (TTS), designed to verify and select the best responses from large language models (LLMs). However, this promise is challenged by recent benchmarks where simple majority voting, which ignores PRM signals, occasionally outperforms standard PRM-based selection. This raises a critical question: How can we effectively utilize verification signals from PRMs for TTS? To address this, we start by developing a theoretical framework for optimally combining signals from both the LLM and the PRM. Our framework reveals that the optimal strategy is a weighted aggregation of responses, a strategy whose effectiveness hinges on estimating weights that capture the complex interplay between the models. Based on our theoretical results, we empirically show that these optimal weighting functions differ significantly across LLM-PRM pairs and, notably, often assign substantial negative weights. Motivated by these insights, we propose efficient pre-computation methods to calibrate these weighting functions. Extensive experiments across 5 LLMs and 7 PRMs demonstrate that our calibration method significantly boosts the TTS efficiency, surpassing the performance of vanilla weighted majority voting while using only $21.3\%$ of the computation. Ultimately, our work demonstrates that investing in a more intelligent aggregation strategy can be a more convincing path to performance gains than simply scaling test-time computation.

## 1 INTRODUCTION

The pursuit of advanced reasoning in Large Language Models (LLMs) has largely been driven by scaling up model size and training data (Ouyang et al., 2022). While effective, this approach entails prohibitive computational costs (Snell et al., 2025). An increasingly popular alternative is Test-Time Scaling (TTS)(Liu et al., 2025; Madaan et al., 2023), a paradigm that enhances the performance of a fixed LLM by allocating more computational resources at inference time. A prominent TTS strategy involves generating a multitude of candidate solutions and then selecting the most promising one. This "generate-and-select" framework relies heavily on the quality of the selection mechanism, which is tasked with identifying the correct response from a pool of diverse, model-generated outputs. The central challenge, therefore, lies in designing a selection strategy that can effectively harness the collective evidence from multiple generated responses to maximize final performance.

To address this selection problem, a common approach is to employ a Process Reward Model (PRM) (Lightman et al., 2024; Li & Li, 2025; Zheng et al., 2024), a sophisticated verifier trained on human feedback to score the quality of reasoning steps. The standard protocol, Best-of-N (BoN), simply selects the answer from the single response that receives the highest PRM score. Intuitively, this should leverage the detailed, step-by-step evaluation capabilities of the PRM. However, this intuition is challenged by a surprising and counter-intuitive empirical reality: on recent benchmarks (Zhang et al., 2025b), the far simpler method of majority voting (Wang et al., 2023), which completely ignores the expensive PRM and relies solely on the

consensus of the LLM's own generations, can outperform PRM-guided BoN. This paradox suggests a fundamental misalignment in how we utilize verifier signals. If a powerful, costly-to-train PRM can be bested by a simple vote count, it implies we are failing to properly integrate its nuanced feedback.

In this work, we dive into the interactions between LLMs and PRMs to find better aggregation of signals from both models for more efficient TTS. We begin by formalizing the task of aggregating responses as a Maximum a Posteriori (MAP) estimation problem, revealing that the optimal aggregation strategy is not to simply pick the best-scoring response, but to perform a weighted majority vote. Interestingly, the optimal weight for each response is a function of two distinct components: a term derived from the PRM's score, reflecting the quality of the reasoning, and a term derived from the LLM's own reliability. This formulation provides a principled framework for unifying the evidence from both the generator and the verifier.

To understand the practical implications of this theoretical result, we conduct an empirical analysis to characterize the optimal weighting function, uncovering two critical insights. First, the shape of the optimal function is highly dependent on the specific LLM-PRM pair, indicating that a one-size-fits-all approach is inherently suboptimal. Second, we find that optimal functions consistently assign *negative* weights to responses with low PRM scores. This reveals a key deficiency in existing methods (Wang et al., 2024): they fail to leverage the negative evidence provided by a low-quality response. A response judged to be poor by the PRM should not simply be ignored; it should actively count *against* its proposed answer.

Motivated by these insights, we introduce simple yet effective calibration methods to learn approximations of these optimal weighting functions from a small, one-time pre-computed dataset. We propose both non-parametric and parametric approaches that explicitly capture the model-specific nature of the weights and incorporate the mechanism of penalizing low-quality responses. Extensive experiments across 5 different LLMs and 7 PRMs on the MATH (Hendrycks et al., 2021) datasets demonstrate the superiority of our approach. Our calibrated weighted voting method consistently outperforms baselines, including standard BoN and vanilla weighted voting. Notably, it achieves higher accuracy than these methods while using approximately 37.1% and 21.3% of the test-time computation, demonstrating a significant improvement in TTS efficiency. In summary, our contributions are:

- We develop a theoretical framework for optimally aggregating LLM generations and PRM scores, demonstrating that the solution is a weighted majority vote combining signals from both models.
- We empirically characterize the optimal weighting function, revealing its model-dependent nature and, interestingly, the importance of assigning negative weights to low-quality responses.
- We propose practical calibration methods to learn these weighting functions, enabling efficient and effective test-time scaling.
- Through extensive experiments, we show that our calibrated aggregation strategy significantly improves TTS efficiency, achieving superior performance with substantially less computational overhead.

## 2 Related work

**Test-Time Scaling.** The pursuit of improved model performance without retraining has led to the paradigm of Test-Time Scaling (TTS), which allocates more computational resources at inference time Zhang et al. (2025a). A dominant strategy within TTS is the "generate-and-select" framework, often formalized as Best-of-N (BoN) sampling, where $N$ candidate solutions are generated and a selection mechanism chooses the best one Ichihara et al. (2025). A foundational method in this area is Self-Consistency (SC), which samples multiple diverse reasoning paths and selecting the final answer via a simple majority vote Wang et al. (2023). The intuition is that an answer derived from multiple independent lines of thought is more likely to be correct. While effective, SC's primary drawback is its high computational cost. This has motivated more efficient variations, such as Confidence-Informed Self-Consistency (CISC), which introduced a weighted majority vote based on the model's self-assessed confidence to reduce the required sample size (Taubenfeld et al.,

2025). In parallel, other approaches use an external verifier, like a PRM, to select the single highest-scoring candidate (Uesato et al., 2022). Our work builds on the idea of weighted voting, but instead of relying on the LLM's self-assessment, we derive weights from a principled theoretical framework that combines both the LLM's consensus signal and the external verifier's scores.

**Reward Modeling.** A crucial component of many TTS strategies is an external verifier, or reward model (RM), trained to score the quality of generated responses. An early, influential work by Cobbe et al. (2021) demonstrated that training a dedicated verifier to select the best solution from many candidates could improve performance on math word problems more effectively than fine-tuning the generator itself Uesato et al. (2022). This spurred a distinction between two supervision strategies: Outcome Reward Models (ORMs), which are trained on the correctness of the final answer, and Process Reward Models (PRMs), which are trained on step-by-step human feedback. An initial comparison by Uesato et al. (2022) found that ORMs could achieve similar final-answer accuracy with less supervision, but PRMs were necessary to ensure the faithfulness of the reasoning process (Zheng et al., 2024). Subsequent work by Lightman et al. (2024) solidified the superiority of PRMs on more challenging tasks, establishing process supervision as a key technique for building reliable verifiers, despite its high annotation cost (Wang et al., 2024). Our work focuses on how to best leverage the signals from these powerful but expensive-to-train PRMs.

## 3 Optimal Response Aggregation

In this section, we aim to explore the optimal aggregation strategy for signals from the LLM and PRM. We start by formalizing this as a Maximum a Posteriori (MAP) estimation problem, and derive an optimal aggregation strategy in Section 3.1. Then, in Section 3.2, we manage to estimate the quantities in the optimal aggregation strategy empirically and offer a few critical insights on the optimal weighting strategy.

### 3.1 Theoretical Analysis of Optimal Response Aggregation

**Problem Setup.** Let $M$ be the LLM and $V$ be the PRM (Verifier). For a single prompt, $M$ generates an ensemble of $L$ responses, $\mathcal{G} = \{g_1, g_2, \ldots, g_L\}$. Each response $g_i$ consists of a reasoning process $r_i$ and a final answer $s_i = f(r_i)$. The PRM $V$ evaluates each generation $g_i$ and produces a scalar score $p_i$. Let $\mathcal{P} = \{p_1, p_2, \ldots, p_L\}$ be the set of these scores. The set of unique candidate answers is $\mathcal{A} = \{\alpha_1, \ldots, \alpha_m\}$. Our objective is to determine the most probable true answer $\hat{\alpha}$ given all available evidence.

We aim to find the answer $\alpha_k$ that maximizes the posterior probability $P(\alpha_k|\mathcal{G}, \mathcal{P}, M, V)$. By Bayes' theorem, and assuming a uniform prior over answers $P(\alpha_k|M, V)$, this is equivalent to maximizing the likelihood of the evidence:

$$\hat{\alpha} = \arg\max_{\alpha_k \in \mathcal{A}} P(\mathcal{G}, \mathcal{P}|\alpha_k, M, V) \tag{1}$$

We can decompose this likelihood into two factors: $P(\mathcal{G}, \mathcal{P}|\alpha_k, M, V) = P(\mathcal{P}|\mathcal{G}, \alpha_k, V) \times P(\mathcal{G}|\alpha_k, M)$. This reflects the causal process: the LLM $M$ generates responses $\mathcal{G}$, and then the Verifier $V$ produces scores $\mathcal{P}$ based on $\mathcal{G}$. To make this tractable, we introduce two conditional independence assumptions:

**Assumption 3.1** (Score and Generation Independence)**.** *The PRM score $p_i$ for a generation $g_i$ is conditionally independent of all other generations, given $g_i$ and the true answer $\alpha_k$. The LLM generations $g_i$ are conditionally independent of each other, given the true answer $\alpha_k$.*

$$P(\mathcal{P}|\mathcal{G}, \alpha_k, V) = \prod_{i=1}^{L} P(p_i|g_i, \alpha_k, V), P(\mathcal{G}|\alpha_k, M) = \prod_{i=1}^{L} P(g_i|\alpha_k, M)$$

With these assumptions, the log-likelihood becomes a sum over individual responses: $\text{LL}(\alpha_k) = \sum_{i=1}^{L} \log P(p_i|g_i, \alpha_k, V) + \sum_{i=1}^{L} \log P(g_i|\alpha_k, M)$. We hypothesize that under the condition $\alpha_k$, a gen-

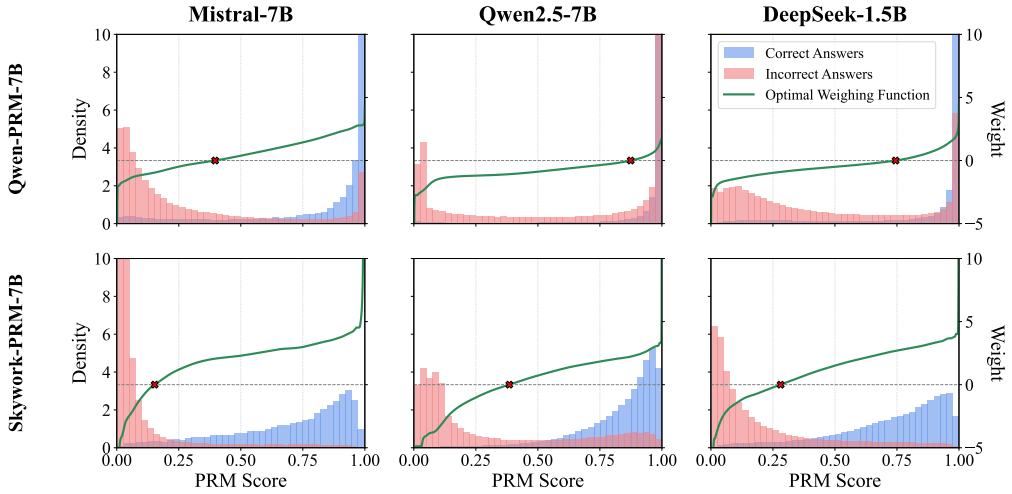


Figure 1: The PRM score distributions and optimal weighing functions on 6 combinations of LLM-PRM pairs. **Left y-axis:** the probability density of the PRM scores. **Right y-axis:** the optimal weights $w^*(p)$ learned via KDE for different LLM-PRM pairs. Note their model-dependent nature and the presence of negative weights for low PRM scores.

eration $g_i$ with answer $s_i = \alpha_k$ is correct ($c_i = 1$), and incorrect ($c_i = 0$) otherwise. The term $P(g_i|\alpha_k, M)$ is simplified to $P(s_i|\alpha_k, M)$, where we assume a simple probability model for the LLM: it produces the correct answer with probability $q_M$ and any specific incorrect answer with probability $(1 - q_M)/(m - 1)$.

**Theorem 3.2** (Optimal Aggregation Score)**.** *Under the assumptions above, maximizing the log-likelihood is equivalent to maximizing the score:*

$$\textit{Score}(\alpha_k) = \sum_{i:s_i=\alpha_k} w_i, \quad \textit{where } w_i = \underbrace{\log \frac{P(p_i|c_i = 1, V)}{P(p_i|c_i = 0, V)}}_{\textit{PRM Signal Term}} + \underbrace{\log \frac{q_M \cdot (m-1)}{1 - q_M}}_{\textit{LLM Signal Term}}$$

*Proof.* The full derivation is in Appendix A. The key insight is that the log-likelihood can be rearranged into a sum of weights for responses voting for $\alpha_k$, plus terms that are constant with respect to $\alpha_k$ and can be dropped from the argmax. The weight $w_i$ for each vote combines signals from two sources: the PRM's score (via the likelihood ratio of the score $p_i$ occurring for correct vs. incorrect reasoning) and the LLM's intrinsic reliability (via the term involving $q_M$), also referred to as question difficulty (Snell et al., 2025). □

### 3.2 Empirical Analysis of the Optimal Weighting Function

**Instantiating the optimal weights**. To instantiate the optimal weight $w_i$ from Theorem 3.2, we perform a *per-question estimation* using the ground-truth labels for the specific set of $L$ responses generated for that question. To estimate the PRM Signal term, we apply a separate Kernel Density Estimation (KDE) on the logit space for each individual question to estimate the score distributions $P(p|c, V)$ on the specific question. For details of our KDE estimation in the logit space, please refer to Section 4.1. To estimate the LLM Signal term, we simply set $q_M$ to be the true accuracy of the $L$ responses for that specific question.

**Characterizing the optimal weighting function**. Taking the PRM's ability to distinguish correct and wrong answers into account, this optimal aggregator provides a much tighter performance upper bound than

Pass@N, demonstrating the potential of our framework, as shown in Figure 2. More importantly, analyzing the structure of the learned weighting function $w^*(p)$ (Figure 1) reveals two critical insights:

- **Weighting functions are highly model-dependent.** The shape of the optimal function varies dramatically depending on the specific LLM and PRM being used. A simple, fixed function (e.g., using the PRM score directly as a weight) is unlikely to be optimal across different model pairs. This underscores the necessity of a calibration procedure tailored to the specific models in use.
- **Presence of negative weights.** An interesting and consistent finding is that low PRM scores are mapped to negative weights. This implies that a response deemed incorrect by the PRM provides strong evidence *against* its proposed answer, and repetition of low-quality responses does not add to the likelihood of their answer being correct. Standard methods like Best-of-N, which only consider the top-scoring candidate, or majority voting, which ignores scores entirely, fail to leverage this powerful negative signal. An effective aggregation strategy must penalize answers supported by low-quality reasoning.

These findings motivate the need for a practical method that can approximate these complex, non-linear, and often negative weighting functions without requiring ground-truth labels at test time. We address this challenge in the following section.

## 4 Practical Calibration Methods

The optimal analysis confirmed the need for a calibrated, model-specific weighting function. We now introduce practical methods to learn these functions using a one-time, pre-computed calibration set $D_{cal} = \{(r_1, p_1, c_1), ..., (r_n, p_n, c_n)\}$. Once a weighting function $w(p)$ is learned, the final answer is selected by a weighted vote: $\hat{\alpha} = \arg\max_{\alpha_k \in \mathcal{A}} \sum_{i:s_i=\alpha_k} w(p_i)$.

### 4.1 Non-parametric Weighting functions

One of the most straightforward ways towards the optimal aggregation strategy is to directly estimate the unknown quantities in the optimal weighting function 3.2, i.e., PRM score distributions $P(p|c = 1, V)$, $P(p|c = 0, V)$, and LLM reliability $q_M$.

**Estimating PRM score distribution.** To capture the nuances of the PRM score distribution on different LLM and PRM combinations, we apply the Kernel Density Estimation (KDE). Compared to other estimation methods, such as histogram estimation or parametric estimations, KDEs are smooth, continuous, and more flexible. However, while the PRM score is within the probability space between 0 and 1, KDEs are not bounded, spilling probability density outside this range. Consequently, we first convert the scores from the probability space to the logit space with the logit function $\text{logit}(p) = \log(\frac{p}{1-p})$. Then, we perform KDE of the scores within the logit space. Specifically, the PRM score distribution is estimated as:

$$\hat{f}_c(p) = \frac{1}{|D_c| \cdot h} \sum_{i \in D_c} K\left(\frac{\text{logit}(p) - \text{logit}(p_i)}{h}\right) \tag{2}$$

where $D_c = \{i|c_i = c\}$ separates responses within the calibration set $D_{cal}$ according to their correctness $c_i$. $K$ and $h$ are the kernel and bandwidth of KDE.

**Estimating LLM reliability.** To estimate $q_M$ at test time without labels, we first train a simple binned probability calibrator $g(\cdot)$ on the PRM scores from the calibration set. During inference, we calculate the calibrated probability for each of the $L$ generated responses $D_{test} = \{(r'_1, p'_1), ..., (r'_L, p'_L)\}$ to the test question and approximate $q_M$ as their average, i.e., $\hat{q}_M = \frac{1}{|D_{test}|} \sum_{i \in D_{test}} g(p'_i)$.

Given the estimations above, according to Equation 3.2, we have the estimated weighting function:

$$w_{KDE}(p) = \log(\hat{f}_1(p)) - \log(\hat{f}_0(p)) + \log(\hat{q}_M) + \log(m-1) - \log(1-\hat{q}_M)$$

This KDE is the practical counterpart to the optimal estimator, where the PRM score distribution is also estimated on $D_{test}$ with additional access to the correctness of test responses $c'_i$.

### 4.2 Parametric Weighting Functions

As an alternative, we explore simpler parametric forms for $w(p)$, optimizing parameters on the calibration set via grid search. These methods are guided by our insight about the importance of a zero-crossing point, controlled by the parameter $b$. This parameter acts as a threshold, making the weight positive for scores above it and negative for scores below, directly implementing the penalization of low-quality responses.

**Logit Weighting.** Inspired by the log-ratio form in our theorem, we model the weight as:

$$w_{logit}(p) = \text{logit}(p) - \text{logit}(b)$$

**Linear Weighting.** As a simpler baseline, we model the weight as:

$$w_{linear}(p) = p - b$$

During grid search, the parameter $b$ is searched within the range $[0, 1]$ and $[-1, 1]$ for Logit Weighted Voting (**Logit WV**) and Linear Weighted Voting (**Linear WV**), respectively.

## 5 Experiments

In this section, we first conduct a comprehensive evaluation of the scaling methods across 35 combinations of LLM and PRM in Section 5.2. Then we dive into the principles of the proposed methods in Section 5.3.

### 5.1 Experimental Setup

**Models.** To capture the complexities of signal aggregation in practice, we use 5 LLMs across 3 model series (Mistral-7B (Wang et al., 2024), Qwen2.5-1.5B/7B (Yang et al., 2024), DeepSeek-1.5B/7B (DeepSeek-AI, 2025)) and 7 PRMs based on Qwen (Qwen2.5-PRM800K (Song et al., 2025), Qwen2.5-PRM-7B (Zhang et al., 2025b), Skywork-PRM-1.5/7B (He et al., 2024)) , Llama (Llama3.1-8B-PRM-Mistral/DeepSeek (Xiong et al., 2024)), and Mistral (math-shepherd-7b-prm (Wang et al., 2024)) series. During generation, we set the top-p and temperature configuration to 0.9 and 0.7, respectively.

**Data.** To simulate the scenarios where the reliability of the LLM signals varies, we evaluate performance on task with various difficulties, the MATH training set (MATH) and test set (MATH500) (Hendrycks et al., 2021). For the MATH dataset, we randomly sample 1k out of 7.5k samples as the calibration set and the rest as the test set. For MATH500, we randomly sample 100 out of 500 samples as the calibration set and the rest as the test set. For the MATH500 dataset, we sample 112 responses from each LLM for each question. For the MATH dataset, we sample 32 responses from each LLM for each question, due to its large size. Then, these collected responses are scored by each of the 7 PRMs. As such, we ensure all the scaling methods are using identical responses and PRM scores for fair comparison.

**Baselines.** We compare our calibrated weighted voting against several methods:

- **Majority Vote**: The answer with the most votes is selected, ignoring PRM scores. This is the standard self-consistency approach. $\hat{\alpha} = \arg\max_{\alpha_k \in \mathcal{A}} \sum_{i=1}^{L} \mathbb{I}(s_i = \alpha_k)$.
- **Best-of-N (BoN)**: The answer from the single response with the highest PRM score is chosen. $\hat{\alpha} = s_{i^*}$ where $i^* = \arg\max_i p_i$.
- **Vanilla Weighted Vote**: A weighted vote where the raw, uncalibrated PRM score $p_i$ is used as the weight. $\hat{\alpha} = \arg\max_{\alpha_k \in \mathcal{A}} \sum_{i:s_i=\alpha_k} p_i$.

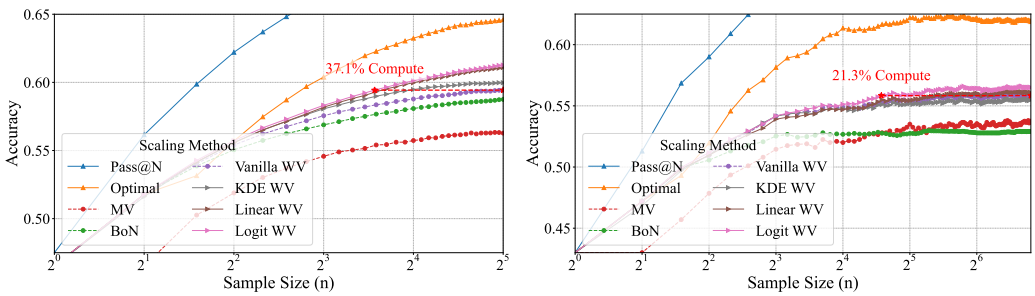


Figure 2: The performance of various scaling methods averaged across all LLM and PRM combinations. The computation efficiency improvement of the Logit WV compared to the best-performing baseline, Vanilla WV, is marked in red. **Left:** On the MATH dataset. **Right:** On the MATH500 dataset.

We also report two theoretical bounds: **Pass@N**, which considers a problem solved if at least one response is correct, and **Optimal** as discussed in Section 3.2.

### 5.2 Main Results

**Weighting function calibration significantly boosts TTS efficiency.** We evaluate the effectiveness of the proposed weighting function calibration methods by performing calibration for each LLM and PRM pair before scaling test-time compute. As shown in Figure 2, calibrating before scaling significantly boosts the efficiency. In particular, on the MATH and MATH500 datasets, the logit-based calibration method surpasses the performance of vanilla weighting voting methods *with approximately 37.1% and 21.3% of compute on average across 35 LLM-PRM pairs*.

For detailed results on LLM-PRM pairs, we show the performance of various scaling methods in Table 1. We can see that calibration methods consistently outperform baseline scaling methods across LLMs and PRMs. Generally, Logit Weighted Voting performs the best in most cases. In particular, on the Llama3.1-Mistral-8B PRM, Logit WV outperforms the best-performing baseline method, Vanilla WV, by 3 points of accuracy (61.2 v.s. 58.2) on average across 5 LLMs, *which would otherwise take an exponential amount of test-time compute to achieve*. This strongly supports our claim of calibrating the weighting function before expensive TTS. Please refer to Appendix B for more detailed results.

### 5.3 Empirical Analysis

**Are negative weights necessary in utilizing the PRM signals?** To further verify our insight that negative weights are necessary for better utilization of the PRM signals, we show the grid search result on the offset parameter $b$ of Logit WV and Linear WV, where, in both cases, its value suggests the zero-crossing point, assigning negative weights to responses whose PRM score is lower than $b$. As shown in Figure 3, for both weighting functions, the optimal offset $b$ is consistently larger than zero across all LLMs, proving the necessity of negative weights in efficient TTS. Furthermore, the zero-crossing points are different for each LLM, but are generally consistent for the same LLM using different weighting functions (Linear and Logit), demonstrating the PRM's varying capability in distinguishing positive and negative responses from different LLMs. This further supports our claim to take the unique interactions between the models into account, i.e., calibration, for efficient TTS.

**What's the remaining gap and challenges towards the optimal weighting function?** To answer this question, we compare our dataset-wise estimation of the PRM score weighting function $\log(\frac{f_1(p)}{f_0(p)})$ with the

Table 1: Accuracy of TTS methods at sample size 32. The best method for each case is in bold.

| PRM | Method | Mistral-7B | Qwen2.5-1.5B | Qwen2.5-7B | DeepSeek-1.5B | DeepSeek-7B | Average |
|---|---|---|---|---|---|---|---|
| Qwen- PRM800K- 7B | Optimal | 53.9 | 64.2 | 70.5 | 57.4 | 57.1 | 60.6 |
| | BoN | 47.6 | 56.7 | 64.0 | 43.1 | 45.2 | 51.3 |
| | MV | 49.1 | 60.0 | 66.6 | 51.8 | 51.3 | 55.8 |
| | Vanilla WV | 50.6 | 61.0 | 66.4 | 51.5 | 51.2 | 56.1 |
| | KDE WV | 50.1 | 60.6 | **66.8** | **51.8** | **51.6** | 56.2 |
| | Linear WV | **51.1** | 61.0 | 66.4 | 51.7 | 51.3 | **56.3** |
| | Logit WV | 49.7 | **61.1** | 66.4 | 51.7 | 51.3 | 56.0 |
| Qwen- PRM- 7B | Optimal | 56.8 | 67.7 | 74.5 | 66.4 | 63.6 | 65.8 |
| | BoN | 51.8 | 62.3 | 69.3 | 61.0 | 59.6 | 60.8 |
| | MV | 49.1 | 60.0 | 66.6 | 51.8 | 51.3 | 55.8 |
| | Vanilla WV | **52.4** | 62.4 | 69.0 | 57.7 | 56.6 | 59.6 |
| | KDE WV | 50.7 | 61.6 | 68.6 | 55.1 | 56.3 | 58.5 |
| | Linear WV | 52.4 | 62.4 | 68.4 | 61.8 | 62.4 | 61.5 |
| | Logit WV | 52.3 | **63.1** | **69.5** | **62.8** | **63.1** | **62.1** |
| Llama3.1- Mistral- 8B | Optimal | 57.1 | 65.6 | 73.4 | 64.3 | 62.2 | 64.5 |
| | BoN | 52.4 | 57.3 | 67.3 | 55.2 | 53.7 | 57.2 |
| | MV | 49.1 | 60.0 | 66.6 | 51.8 | 51.3 | 55.8 |
| | Vanilla WV | 51.8 | 61.7 | 68.3 | 54.7 | 54.4 | 58.2 |
| | KDE WV | 51.6 | 61.6 | 67.5 | 55.1 | 57.1 | 58.6 |
| | Linear WV | 53.1 | 62.2 | 68.2 | **60.1** | 60.3 | 60.8 |
| | Logit WV | **53.7** | **62.4** | **69.0** | 60.1 | **60.7** | **61.2** |
| Llama3.1- DS- 8B | Optimal | 56.4 | 65.7 | 73.0 | 64.2 | 62.0 | 64.2 |
| | BoN | 49.7 | 58.6 | 67.5 | 57.6 | 55.7 | 57.8 |
| | MV | 49.1 | 60.0 | 66.6 | 51.8 | 51.3 | 55.8 |
| | Vanilla WV | 52.4 | **61.2** | 68.6 | 54.8 | 55.4 | 58.5 |
| | KDE WV | 50.3 | 60.7 | 67.5 | 54.0 | 53.8 | 57.3 |
| | Linear WV | **52.8** | 61.1 | 68.6 | 59.1 | 58.8 | 60.1 |
| | Logit WV | 52.7 | 61.1 | **68.8** | **60.0** | **59.6** | **60.4** |
| Skywork- PRM- 1.5B | Optimal | 55.9 | 72.0 | 74.1 | 66.0 | 68.3 | 67.3 |
| | BoN | 52.2 | 68.9 | **71.4** | 62.2 | **65.2** | 64.0 |
| | MV | 49.1 | 64.4 | 66.6 | 51.8 | 55.8 | 57.5 |
| | Vanilla WV | 52.7 | 68.2 | 70.5 | 58.3 | 60.6 | 62.1 |
| | KDE WV | 51.7 | 66.3 | 69.8 | 57.3 | 59.3 | 60.9 |
| | Linear WV | 53.5 | 68.8 | 70.8 | **64.6** | 64.7 | 64.5 |
| | Logit WV | **53.6** | **69.1** | 70.8 | 64.4 | 64.8 | **64.6** |

optimal per-question estimation on several questions. As shown in the left subplot of Figure 4, while our estimation captures the dataset-wise PRM score weighting function, the optimal weighting function for each individual question varies largely, suggesting a global scoring function is still suboptimal. We also examine how accurate our estimation of the LLM reliability term $q_M$ is with Mean Absolute Error. As shown in the right subplot of Figure 4, while compared to a fixed global LLM reliability, using calibrated PRM scores to estimate this term effectively reduces the error, the error for most PRMs is still larger than 0.2, which is far from negligible. This also explains the relatively low performance of KDE WV compared to the parametric counterparts. In conclusion, accurately estimating either the PRM or the LLM part of the weight requires nuanced estimation for individual questions, explaining why the non-parametric KDE estimation underperforms the parametric ones. We find such per-question estimation inherently difficult in our attempts to learn a meta model to predict per-question weighting functions, which struggles to fit and generalize.

**How does calibration set size affect the performance?** We rearrange the split of the MATH dataset to reserve 5k questions as the test, and the rest as the pool of calibration data. As shown in the right subplot

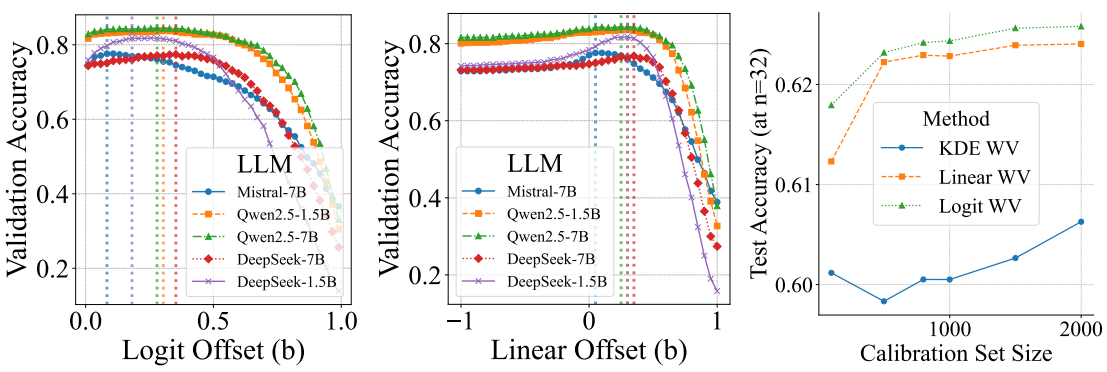


Figure 3: **Left and Middle:** The grid search result of the offset parameter $b$ for both Logit WV and Linear WV, where the optimal value is marked with vertical lines. The consistently positive optimal value across LLMs demonstrates the necessity of negative weights. **Right:** The performance of the calibration methods when we scale the calibration set size. The performance can be further improved with larger calibration sets.

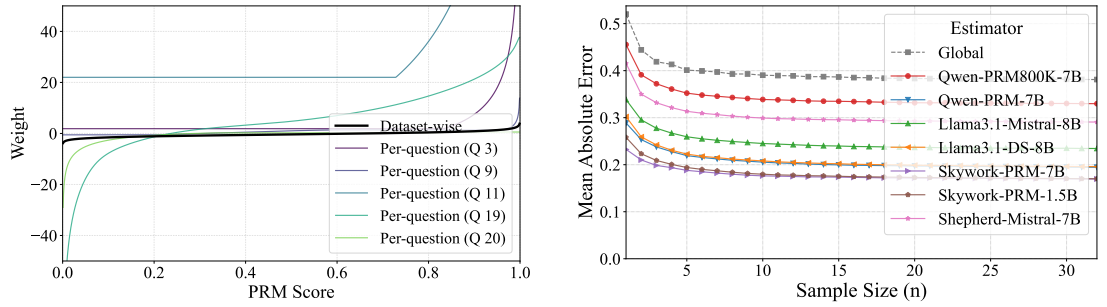


Figure 4: **Left:**The comparison of the dataset-wise estimated PRM score weighting function and the per-question estimated optimal PRM score weighting function. A large variance among questions can be seen. **Right:** The mean absolute error of estimated $\hat{q}_M$ compared to the true $q_M$.

of Figure 3, while the validation set used in the main experiments is relative small, the performances of the calibration methods can be further enhanced if we scale the calibration set size to calibrate the weighting function better.

## 6 Conclusion

We address the suboptimal use of Process Reward Models (PRMs) in Test-Time Scaling (TTS). Through a theoretical MAP framework, we show that the optimal aggregation strategy is a weighted majority vote combining signals from both the LLM and PRM. Empirically, we find these optimal weights are model-dependent and, critically, assign large negative values to penalize low-quality responses—a powerful signal neglected by standard methods. We propose simple calibration methods to learn these functions. Our calibrated weighted voting boosts TTS efficiency, achieving superior accuracy over baselines like Best-of-N with approximately 37.1% and 21.3% computational cost. This work demonstrates that intelligent aggregation is a more efficient path to performance gains than simply scaling test-time compute.

## REFERENCES


Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168. arXiv: 2110.14168 [cs.LG].

DeepSeek-AI. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, 2025. URL https://arxiv.org/abs/2501.12948. _eprint: 2501.12948.

Jujie He, Tianwen Wei, Rui Yan, Jiacai Liu, Chaojie Wang, Yimeng Gan, Shiwen Tu, Chris Yuhao Liu, Liang Zeng, Xiaokun Wang, Boyang Wang, Yongcong Li, Fuxiang Zhang, Jiacheng Xu, Bo An, Yang Liu, and Yahui Zhou. Skywork-o1 Open Series, November 2024. URL https://doi.org/10.5281/zenodo.16998085. Version Number: 1.0.0.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring Mathematical Problem Solving With the MATH Dataset. *NeurIPS*, 2021.

Yuki Ichihara, Yuu Jinnai, Tetsuro Morimura, Kaito Ariu, Kenshi Abe, Mitsuki Sakamoto, and Eiji Uchibe. Evaluation of Best-of-N Sampling Strategies for Language Model Alignment, February 2025. URL http://arxiv.org/abs/2502.12668. arXiv:2502.12668 [cs].

Wendi Li and Yixuan Li. Process reward model with q-value rankings. In *The thirteenth international conference on learning representations*, 2025. URL https://openreview.net/forum?id=wQEdh2cgEk.

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

Fan Liu, Wenshuo Chao, Naiqiang Tan, and Hao Liu. Bag of Tricks for Inference-time Computation of LLM Reasoning, 2025. URL https://arxiv.org/abs/2502.07191. _eprint: 2502.07191.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative Refinement with Self-Feedback, May 2023. URL http://arxiv.org/abs/2303.17651. arXiv:2303.17651 [cs].

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Proceedings of the 36th international conference on neural information processing systems*, Nips '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 978-1-71387-108-8. Number of pages: 15 Place: New Orleans, LA, USA tex.articleno: 2011.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling test-time compute optimally can be more effective than scaling LLM parameters. In *The thirteenth international conference on learning representations*, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. PRMBench: A Fine-grained and Challenging Benchmark for Process-Level Reward Models, 2025. URL https://arxiv.org/abs/2501.03124. _eprint: 2501.03124.

Amir Taubenfeld, Tom Sheffer, Eran Ofek, Amir Feder, Ariel Goldstein, Zorik Gekhman, and Gal Yona. Confidence Improves Self-Consistency in LLMs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20090–20111, 2025. doi: 10.18653/v1/2025.findings-acl.1030. URL http://arxiv.org/abs/2502.06233. arXiv:2502.06233 [cs].

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, 2022. URL https://arxiv.org/abs/2211.14275. arXiv: 2211.14275 [cs.LG].

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 Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 9426–9439, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

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

Wei Xiong, Hanning Zhang, Nan Jiang, and Tong Zhang. An Implementation of Generative PRM, 2024. URL https://github.com/RLHFlow/RLHF-Reward-Modeling. Publication Title: GitHub repository.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-Math Technical Report: Toward Mathematical Expert Model via Self-Improvement. *arXiv preprint arXiv:2409.12122*, 2024.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, Irwin King, Xue Liu, and Chen Ma. A Survey on Test-Time Scaling in Large Language Models: What, How, Where, and How Well?, May 2025a. URL http://arxiv.org/abs/2503.24235. arXiv:2503.24235 [cs].

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning, 2025b. URL https://arxiv.org/abs/2501.07301. arXiv: 2501.07301 [cs.CL].

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. ProcessBench: Identifying process errors in mathematical reasoning, 2024. URL https://arxiv.org/abs/2412.06559. arXiv: 2412.06559 [cs.AI].

# A PROOFS

## A.1 DERIVATION OF THEOREM 3.2

Our objective is to find the answer $\hat{\alpha}$ that maximizes the posterior probability $P(\alpha_k|\mathcal{G}, \mathcal{P}, M, V)$. Assuming a uniform prior over answers, this is equivalent to maximizing the log-likelihood, $\text{LL}(\alpha_k)$. From the main text, the log-likelihood under Assumption 3.1 is:

$$\text{LL}(\alpha_k) = \sum_{i=1}^{L} \log P(p_i|g_i, \alpha_k, V) + \sum_{i=1}^{L} \log P(g_i|\alpha_k, M) \tag{3}$$

Let $c_i \in \{0, 1\}$ be a binary variable indicating whether generation $g_i$ is correct ($c_i = 1$) or incorrect ($c_i = 0$). Under the hypothesis that the true answer is $\alpha_k$, the correctness of $g_i$ is determined by its answer $s_i$. Specifically, $c_i = 1$ if $s_i = \alpha_k$, and $c_i = 0$ if $s_i \neq \alpha_k$.

We can now analyze the two components of the log-likelihood separately.

**Part 1: The PRM Signal Term**

The first sum can be split based on whether a generation's answer $s_i$ matches the candidate answer $\alpha_k$:

$$\sum_{i=1}^{L} \log P(p_i|g_i, \alpha_k, V) = \sum_{i:s_i=\alpha_k} \log P(p_i|g_i, c_i = 1, V) + \sum_{i:s_i\neq\alpha_k} \log P(p_i|g_i, c_i = 0, V) \tag{4}$$

To isolate the terms relevant to the maximization over $\alpha_k$, we rewrite the second sum by noting that $\sum_{i:s_i\neq\alpha_k}(\cdot) = \sum_{i=1}^{L}(\cdot) - \sum_{i:s_i=\alpha_k}(\cdot)$:

$$\begin{aligned} &= \sum_{i:s_i=\alpha_k} \log P(p_i|g_i, c_i = 1, V) + \sum_{i=1}^{L} \log P(p_i|g_i, c_i = 0, V) - \sum_{i:s_i=\alpha_k} \log P(p_i|g_i, c_i = 0, V) \\ &= \sum_{i:s_i=\alpha_k} (\log P(p_i|g_i, c_i = 1, V) - \log P(p_i|g_i, c_i = 0, V)) + \sum_{i=1}^{L} \log P(p_i|g_i, c_i = 0, V) \end{aligned} \tag{5}$$

The second term, $\sum_{i=1}^{L} \log P(p_i|g_i, c_i = 0, V)$, is a sum over all $L$ generations. Since this term does not depend on the choice of the candidate answer $\alpha_k$, it is a constant with respect to our maximization problem and can be dropped. This leaves us with the $\alpha_k$-dependent part of the PRM signal:

$$\text{PRM\_Term}(\alpha_k) = \sum_{i:s_i=\alpha_k} \log \frac{P(p_i|g_i, c_i = 1, V)}{P(p_i|g_i, c_i = 0, V)} \tag{6}$$

**Part 2: The LLM Signal Term**

Next, we analyze the LLM term, simplifying $P(g_i|\alpha_k, M)$ to $P(s_i|\alpha_k, M)$. We use the model's probabilities: $P(s_i = \alpha_k|\alpha_k, M) = q_M$ and, for any $s_j \neq \alpha_k$, $P(s_j|\alpha_k, M) = (1 - q_M)/(m - 1)$. Let $N_k$ be the

count of generations where the answer is $\alpha_k$, i.e., $N_k = |\{i|s_i = \alpha_k\}|$.

$$\begin{aligned}\sum_{i=1}^{L} \log P(s_i|\alpha_k, M) &= \sum_{i:s_i=\alpha_k} \log P(s_i = \alpha_k|\alpha_k, M) + \sum_{i:s_i\neq\alpha_k} \log P(s_i \neq \alpha_k|\alpha_k, M) \\ &= N_k \log q_M + (L - N_k) \log \frac{1-q_M}{m-1} \\ &= N_k \log q_M + L \log \frac{1-q_M}{m-1} - N_k \log \frac{1-q_M}{m-1} \\ &= N_k \left(\log q_M - \log \frac{1-q_M}{m-1}\right) + L \log \frac{1-q_M}{m-1} \end{aligned} \tag{7}$$

Similar to the PRM term, the second part, $L \log \frac{1-q_M}{m-1}$, does not depend on the specific candidate answer $\alpha_k$ (as $q_M, L, m$ are fixed for a given question) and can be dropped from the objective function. The remaining term is:

$$\text{LLM\_Term}(\alpha_k) = N_k \log \frac{q_M \cdot (m-1)}{1-q_M} = \sum_{i:s_i=\alpha_k} \log \frac{q_M \cdot (m-1)}{1-q_M} \tag{8}$$

**Part 3: Combining the Terms**

Maximizing $\text{LL}(\alpha_k)$ is equivalent to maximizing the sum of the $\alpha_k$-dependent terms we derived. Let this new objective function be $\text{Score}(\alpha_k)$:

$$\begin{aligned}\text{Score}(\alpha_k) &= \text{PRM\_Term}(\alpha_k) + \text{LLM\_Term}(\alpha_k) \\ &= \sum_{i:s_i=\alpha_k} \log \frac{P(p_i|c_i = 1, V)}{P(p_i|c_i = 0, V)} + \sum_{i:s_i=\alpha_k} \log \frac{q_M \cdot (m-1)}{1-q_M} \\ &= \sum_{i:s_i=\alpha_k} \left(\log \frac{P(p_i|c_i = 1, V)}{P(p_i|c_i = 0, V)} + \log \frac{q_M \cdot (m-1)}{1-q_M}\right) \end{aligned} \tag{9}$$

This is a weighted majority vote, where the final score for an answer $\alpha_k$ is the sum of weights $w_i$ for all generations $g_i$ that produced that answer. The weight for each generation is:

$$w_i = \underbrace{\log \frac{P(p_i|c_i = 1, V)}{P(p_i|c_i = 0, V)}}_{\text{PRM Signal Term}} + \underbrace{\log \frac{q_M \cdot (m-1)}{1-q_M}}_{\text{LLM Signal Term}}$$

This completes the proof.

# B Additional experiment results

## B.1 Detailed results on the MATH500 dataset

We show the detailed results on each LLM-PRM pair on the MATH500 dataset in Table 2.

# C Limitations and Future Work

**Limitations.** Our work, while demonstrating significant gains, has several limitations. First, our theoretical framework relies on conditional independence assumptions which are simplifications of the complex

Table 2: Accuracy of Aggregation Methods at Sample Size n=112

| PRM | Method | Mistral-7B | Qwen2.5-1.5B | Qwen2.5-7B | DeepSeek-1.5B | DeepSeek-7B | Average |
|---|---|---|---|---|---|---|---|
| Qwen- PRM800K- 7B | Optimal | 37.3 | 68.0 | 69.7 | 58.3 | 62.7 | 59.2 |
| | BoN | 33.0 | 57.0 | 62.0 | 40.3 | 49.0 | 48.3 |
| | MV | 29.3 | 61.3 | 66.0 | 52.7 | **57.0** | 53.3 |
| | Vanilla WV | 31.7 | **63.7** | 65.7 | **53.0** | 56.7 | 54.1 |
| | KDE WV | 30.0 | 62.0 | 66.0 | 53.0 | 57.0 | 53.6 |
| | Linear WV | **34.3** | 63.7 | **66.3** | 52.7 | 57.0 | **54.8** |
| | Logit WV | 32.0 | 63.3 | 66.3 | 52.7 | 56.7 | 54.2 |
| Qwen- PRM- 7B | Optimal | 44.3 | 70.3 | 73.0 | 67.7 | 67.7 | 64.6 |
| | BoN | **40.0** | 63.7 | 67.3 | 59.7 | 65.0 | 59.1 |
| | MV | 29.3 | 61.3 | 66.0 | 52.7 | 57.0 | 53.3 |
| | Vanilla WV | 35.3 | **64.0** | 67.3 | 57.7 | 62.3 | 57.3 |
| | KDE WV | 34.3 | 63.7 | 67.0 | 56.3 | 62.3 | 56.7 |
| | Linear WV | 36.0 | 64.0 | 68.0 | **63.3** | **65.7** | 59.4 |
| | Logit WV | 38.0 | 64.0 | **68.3** | 63.3 | 65.3 | **59.8** |
| Llama3.1- Mistral- 8B | Optimal | 44.0 | 65.3 | 69.3 | 63.0 | 66.0 | 61.5 |
| | BoN | **32.7** | 50.0 | 59.7 | 49.3 | 55.3 | 49.4 |
| | MV | 29.3 | **61.3** | 66.0 | 52.7 | 57.0 | 53.3 |
| | Vanilla WV | 30.0 | 60.0 | **67.0** | 53.7 | 58.7 | 53.9 |
| | KDE WV | 29.0 | 60.7 | 66.0 | 54.3 | 57.7 | 53.5 |
| | Linear WV | 31.3 | 60.0 | 66.0 | **56.0** | **60.0** | **54.7** |
| | Logit WV | 31.7 | 60.3 | 65.7 | 56.0 | 59.7 | 54.7 |
| Llama3.1- DS- 8B | Optimal | 39.0 | 67.7 | 69.7 | 61.3 | 66.0 | 60.7 |
| | BoN | 27.0 | 51.0 | 62.3 | 51.3 | 59.0 | 50.1 |
| | MV | 29.3 | **61.3** | 66.0 | 52.7 | 57.0 | 53.3 |
| | Vanilla WV | **30.0** | 59.3 | **66.3** | 55.3 | 60.0 | 54.2 |
| | KDE WV | 28.7 | 61.3 | 66.0 | 55.3 | 59.7 | 54.2 |
| | Linear WV | 29.3 | 61.0 | 66.3 | 55.7 | **62.0** | **54.9** |
| | Logit WV | 29.3 | 59.7 | 66.3 | **57.7** | 61.3 | 54.9 |
| Skywork- PRM- 1.5B | Optimal | 40.7 | 66.3 | 71.0 | 63.3 | 66.3 | 61.5 |
| | BoN | **38.3** | 56.7 | 63.3 | 54.0 | 58.7 | 54.2 |
| | MV | 29.3 | 61.3 | 66.0 | 52.7 | 57.0 | 53.3 |
| | Vanilla WV | 35.7 | **61.7** | **68.3** | 58.0 | 60.3 | 56.8 |
| | KDE WV | 32.7 | 61.3 | 67.3 | 55.7 | 58.0 | 55.0 |
| | Linear WV | 38.3 | 61.3 | 68.0 | 60.7 | **61.7** | **58.0** |
| | Logit WV | 37.7 | 61.7 | 66.0 | **61.0** | 61.3 | 57.5 |

dependencies between generated responses. Second, our proposed calibration methods learn a single, global weighting function. As our analysis in Section 4.3 shows, the truly optimal function varies on a per-question basis, and our attempts to learn a meta-model to predict these per-question functions were unsuccessful, indicating this is a non-trivial challenge. Finally, while effective, our calibration methods require a small, one-time labeled dataset, and our evaluation has been focused on the domain of mathematical reasoning.

**Future Work.** These limitations point to several promising avenues for future research. The primary challenge is to bridge the gap between global and per-question optimal weighting. Developing methods that can adapt the weighting function at test time based on question-specific features or initial response characteristics could yield further performance gains. Another direction is to explore the generalization of our calibrated aggregation framework to other domains beyond mathematics and to other TTS paradigms, such as sequential refinement or tree-of-thoughts search. Lastly, investigating semi-supervised or unsupervised calibration techniques could reduce the reliance on labeled data, making the approach more accessible and scalable.

**Ethics Statement.** This research aims to improve the computational efficiency of large language models, a goal with positive ethical implications. By achieving higher performance with less computational cost, our methods can contribute to reducing the energy consumption and environmental impact associated with deploying state-of-the-art AI systems. The datasets and models used in this work are standard, publicly available benchmarks and open-source models widely used by the research community. Our work does not involve human subjects, nor does it introduce new capabilities that would increase the risk of misuse of language models. On the contrary, by developing a more nuanced understanding of how to verify and aggregate machine-generated reasoning, this research could contribute to making LLMs more reliable and less prone to generating confident but incorrect outputs. All authors have read and adhered to the ICLR Code of Ethics.

**Reproducibility Statement.** We are committed to ensuring the reproducibility of our work. All LLMs and PRMs used are publicly available models, and we provide details of these models in Section 5.1. The experiments were conducted on the MATH dataset, a standard public benchmark. Our data splitting procedure for calibration and testing is described in Section 5.1. The full theoretical derivation for Theorem 3.2 is provided in Appendix A.1. Key implementation details for our proposed calibration methods, including the KDE procedure and the grid search ranges for parametric models, are described in Section 4.

**LLM Usage.** We use LLMs to polish the writing of this paper, including identifying spelling, grammar mistakes. All suggestions from the LLM are verified by the authors before being incorporated into the paper.