UNCERTAINTY-AWARE STEP-WISE VERIFICATION WITH GENERATIVE REWARD MODELS

Zihuiwen Ye[1]*, **Luckeciano Carvalho Melo**[1]†, **Younesse Kaddar**[1]†,

Phil Blunsom[2], **Sam Staton**[1], **Yarin Gal**[1]

[1]University of Oxford, [2]Cohere

ABSTRACT

Complex multi-step reasoning tasks, such as solving mathematical problems, remain challenging for large language models (LLMs). While outcome supervision is commonly used, process supervision via process reward models (PRMs) provides intermediate rewards to verify step-wise correctness in solution traces. However, as proxies for human judgment, PRMs suffer from reliability issues, including susceptibility to reward hacking. In this work, we propose leveraging uncertainty quantification (UQ) to enhance the reliability of step-wise verification with generative reward models for mathematical reasoning tasks. We introduce CoT Entropy, a novel UQ method that outperforms existing approaches in quantifying a PRM's uncertainty in step-wise verification. Our results demonstrate that incorporating uncertainty estimates improves the robustness of judge-LM PRMs, leading to more reliable verification.

1 INTRODUCTION

Large Language Models (LLMs) have shown impressive reasoning abilities on complex tasks by producing step-by-step solutions in a chain-of-thought (CoT) format (Wei et al., 2022). A common approach to further improve these capabilities is to fine-tune LLMs using reinforcement learning (RL) on generated CoT outputs (Pang et al., 2024). However, applying RL to reasoning tasks introduces a new challenge: credit assignment. In *outcome-only* reward settings, where rewards are assigned only based on the final answer, the model may receive a high reward even when its intermediate steps are flawed. This results in false positives (Zhang et al., 2024), where correct answers are reached by spurious reasoning, and erodes the trust in the reasoning process (Zhang et al., 2025).

To address this issue, recent work has proposed *process supervision* with Process Reward Models (PRMs) that provide feedback at each reasoning step, offering more fine-grained reward signals than simply judging a final answer (Lightman et al., 2023; Wang et al., 2024; Uesato et al., 2022). Benchmarks like PROCESSBENCH (Zheng et al., 2024) aim to evaluate PRMs' ability to discern step-wise correctness in multi-step solution traces, and while there has been promising progress, two major problems still limit the effectiveness of PRMs. First, obtaining high-quality step-by-step annotations is challenging: current efforts relying on human annotation (Lightman et al., 2023), Monte Carlo sampling (Wang et al., 2024), or LLM-as-a-judge (Zheng et al., 2023) are either costly or noisy. Second, since PRMs are ultimately a learned proxy, they are susceptible to over-optimization and reward hacking: a policy may learn to exploit imperfections in the reward model to get high reward without truly improving its reasoning (DeepSeek-AI, 2025).

In this paper, we argue that *uncertainty quantification* (UQ) offers a principled way to improve the reliability of these generative reward models. By detecting when the verifier itself is uncertain at a given step, we can better guard against erroneous feedback and improve the interpretability of the reward signals. While recent work has explored UQ for LLMs on simpler tasks, such as question answering (Shorinwa et al., 2024), to our knowledge, little is known about how to apply UQ in complex, multi-step reward modeling, where errors may arise at any intermediate step. We take

[*] Correspondence to `zihuiwen.ye@cs.ox.ac.uk`.

[†] Equal contribution.

a first step toward closing this gap by proposing an uncertainty-aware verification framework on mathematical reasoning tasks. Our contributions are summarized as follows:

- We propose a novel UQ method for step-wise verification in generative reward models, leveraging CoT-based entropy to capture uncertainty in the verifier's reasoning.
- We show that incorporating uncertainty estimates leads to more robust verification performance for reward models as judges on multi-step reasoning tasks.
- We compare our method with existing UQ methods and show better performance as evaluated by metrics such as AUC, rejection-F1, and selective performance score.
- We offer insights into the main sources of PRM errors in the context of verifying multi-step math solutions by decomposing predictive uncertainty into knowledge uncertainty and aleatoric uncertainty.

2 Problem Statement

2.1 Background on LLM Reasoning and Reward Verification

We follow the standard setup in LLM-based reasoning. Given a LLM representing a policy π, referred to as 'generator', and an input question Q, π generates an output sequence $\mathbf{s}(s_1, s_2, \cdots, s_K)$ by autoregressively sampling tokens from $\pi(- \mid Q)$. For every $t\{1, \ldots, K\}$, let $\mathbf{s}_{<t}$ (resp. $\mathbf{s}_t$) be the subsequence $(s_1, \ldots, s_{t-1})$ (resp. $(s_1, \ldots, s_t)$), and let $\mathbf{s}_{<1} = \mathbf{s}_0$ be the empty sequence. The conditional probability of generating an output sequence $\mathbf{s}$ is given by:

$$\pi(\mathbf{s} \mid Q) = \prod_{t=1}^{K} \pi(s_t \mid Q, \mathbf{s}_{<t}). \tag{1}$$

In the context of reasoning tasks, the sequence $\mathbf{s}(s_1, s_2, \cdots, s_K)$ represents a reasoning path, where s_t represents the step at time t (for $1 \leq t \leq K$) and K is the total number of steps in $\mathbf{s}$. The final answer derived from this reasoning path $\mathbf{s}$ is denoted by $a_{\mathbf{s}}$. Typically, each question Q has a ground truth final correct answer a_Q. The model is considered to have correctly answered Q if its final conclusion $a_{\mathbf{s}}$ matches a_Q.

Given a reasoning path $\mathbf{s}$ generated by a policy $\pi(- \mid Q)$, verifiers or reward models (RMs) are commonly used to evaluate the quality of $\mathbf{s}$ by assigning a reward score $r(Q, \mathbf{s})$. Once trained, RMs can be deployed to either update π in the RL training process or to guide inference-time search. Specifically, given a question Q and a reasoning path $\mathbf{s}$, a verifier parameterized by θ produces a reward score $r_\theta(Q, \mathbf{s})$ that estimates the correctness of $\mathbf{s}$. This reward can depend solely on the final derived answer $a_{\mathbf{s}}$ (as in outcome reward models, ORMs) or on the entire reasoning trace $\mathbf{s}$ (as in process reward models, PRMs). In this work, we focus on PRMs. PRMs are step-wise verifiers that assess the correctness of each step s_t in $\mathbf{s}$ (Lightman et al., 2023; Wang et al., 2024; Uesato et al., 2022), allowing for more fine-grained supervision over the reasoning process than ORMs. They mitigate false positives, where a path leads to the correct solution a_Q but through flawed reasoning (Zhang et al., 2024; 2025). However, one key challenge with developing PRMs lies in the complex and costly process of annotating step-wise labels, leading to less robust training.

2.2 Step-wise Verification with Generative PRM

In this work, we focus on generative PRMs (Lightman et al., 2023), which are trained through a next-token prediction objective that maximizes the likelihood of special tokens indicating step correctness. Formally, we define a random variable with two classes $E_t \in \{0, 1\}$ at each step s_t to indicate whether the step contains an error, where $E_t = 0$ means the step is error-free (correct), and $E_t = 1$ means the step contains an error. For a generative PRM parameterized by θ, the model outputs a predictive distribution $p_\theta\big(E_t \mid Q, \mathbf{s}_t\big)$ at each time step t, reflecting its belief about whether step s_t is correct or incorrect. From this distribution, we define the step-wise reward r_t as the probability that $E_t = 0$:

$$r_t r_\theta(Q, s_t) = p_\theta(E_t = 0 \mid Q, \mathbf{s}_t) = p_\theta\big([\texttt{no\_error}] \mid Q, \mathbf{s}_t\big), \tag{2}$$

where [no_error] is a special token denoting that the step contains no error. By extension, the solution-level reward is given by the product of these step-wise probabilities:

$$r_\theta(Q, \mathbf{s}) = \prod_{t=1}^{K} r_\theta(Q, s_t) = \prod_{t=1}^{K} p_\theta(E_t = 0 \mid Q, s_t). \tag{3}$$

Finally, to obtain a discrete prediction for each step $t\{1, \ldots, K\}$, we select the most likely class from the predictive distribution:

$$\hat{e}_t = \arg\max_{e \in \{0,1\}} p_\theta\big(E_t = e \mid Q, \mathbf{s}_t\big), \tag{4}$$

producing a sequence prediction $\hat{\mathbf{e}} = (\hat{e}_1, \ldots, \hat{e}_K) \in \{0,1\}^K$ for the reasoning trace $\mathbf{s} = (s_1, \ldots, s_K)$. This is then compared to the ground truth sequence labels $\mathbf{y} = (y_1, y_2, \ldots, y_K) \in \{0,1\}^K$. We define the step-wise verification accuracy with an indicator function $(\hat{e}_t = y_t)$, which is 1 if the predicted value equals the ground truth and 0 otherwise. In practice, datasets are annotated in such a way that the reasoning trace terminates at the first error (i.e., at the first index t such that s_t has a ground truth label of 1). Consequently, if we assume a constant error rate at each step, the index of the first positive ground-truth label follows a geometric distribution.

2.3 Uncertainty Quantification with Judge-LM

We now introduce the task of uncertainty quantification for step-wise verification. In the following, we will use an LLM with parameterized policy p_θ (denoting its weights) as a generative PRM, referred to as a 'judge-LM'. Following the LLM-as-a-judge setup (Zheng et al., 2023), we use a prompt $\mathbf{I}$ that asks for the model's verification judgment at each step s_t, such as "`Does this step contain an error?`". Let $\tau(e)$ denote the token corresponding to the label $e \in \{0, 1\}$, for instance, $\tau(0) = \texttt{no}$ and $\tau(1) = \texttt{yes}$. We want to quantify the uncertainty in the judge-LM's decision, and employ methods to detect when the LM is likely to make a mistake. To achieve this, we derive *uncertainty estimates* $u_t g(p_\theta(E_t = e \mid Q, \mathbf{s}_t, \mathbf{I}))$ from the predictive distribution p_θ, where $g(\cdot)$ is an uncertainty-mapping function. Specifically, p_θ over the binary variable E_t is obtained by:

$$p_\theta\big(E_t = e \mid Q, \mathbf{s}_t, \mathbf{I}\big) \;=\; \frac{p_\theta(\tau(e) \mid Q, \mathbf{s}_t, \mathbf{I}))}{\sum\limits_{e' \in \{0,1\}} p_\theta(\tau(e') \mid Q, \mathbf{s}_t, \mathbf{I}))}, \quad e \in \{0,1\}, \tag{5}$$

where the numerator is the token probability assigned by the LM to the token $\tau(e)$. Thus, we obtain a well-defined normalized probability distribution over the possible label outcomes $e \in \{0, 1\}$.

In the following sections, we will investigate how well different uncertainty estimates u_t align with the model's actual correctness at each step (given ground-truth labels in $\{0,1\}^K$), paving the way for uncertainty-aware verification.

3 Chain-of-Thought Entropy

Since a better-calibrated predicted probability distribution p_θ can yield more reliable confidence estimates u (Guo et al., 2017; Cattelan & Silva, 2023), we first improve the estimation given by p_θ with chain-of-thought (CoT) prompting (Wei et al., 2022), which is shown to improve performance in a wide range of tasks. Since verification involves nuanced reasoning, judge-LM verifiers naturally benefit from CoT, which can generate intermediate rationales or critiques to help identify subtle errors that might otherwise be missed by direct verifiers before deciding on the correctness of a solution (Ye et al., 2024; Zhang et al., 2024). In a step-wise verification setting, CoT ensures that reasoning remains focused on assessing the local logical consistency of each atomic step s_t with a binary decision while improving interpretability by providing step-wise feedback on the mistakes. Specifically, we design a prompt $\mathbf{I}_{\text{CoT}}$ (shown in App. A.1) that asks the model to first output a rationale c, and then output an evaluation e of whether the step contains an error.

Our method is inspired by semantic entropy (SE) (Farquhar et al., 2024), which addresses the fact that one concept can be syntactically expressed in multiple ways by computing uncertainty at the level of semantic outcomes rather than specific sequences of words. For example, in response to the question "*What's the capital of France?*", both "*It's Paris.*" and "*Paris.*" convey the same underlying meaning. Concretely, in a QA setting, SE is implemented by sampling and grouping semantically equivalent answer phrases into clusters using NLI-based methods, aggregating their probabilities, and computing entropy over these semantic clusters. However, in a reasoning setting, SE is not directly applicable, as semantic entailment is not clearly defined for reasoning chains, which are more nuanced and open-ended.

So for our step-wise verification task, we adapt Semantic Entropy's core insight while taking a different implementation approach. Rather than clustering semantically equivalent answers with NLI models to determine bidirectional entailment, we leverage the binary structure of verification decisions ($e \in \{0,1\}$) to group diverse reasoning paths by their outcomes, thus avoiding the complex challenge of determining semantic equivalence between mathematical reasoning chains, while still giving reasonable estimates of uncertainty over equivalence classes of reasoning paths.

We call our method 'CoT Entropy': it aims to estimate the uncertainty of the judge-LM over its step-wise verification judgments, conditioned on diverse reasoning paths. Intuitively, our method samples several reasoning paths leading to binary judgments for each step (whether this step is deemed incorrect or not), and then marginalizes all these reasoning paths to compute an estimate of the entropy over the verification judgment.

To simplify the notations, we define $\mathbf{x}_t(Q, \mathbf{s}_t, \mathbf{I}_{\text{CoT}})$, which represents the context up to step t, including the input question Q and the CoT prompt $\mathbf{I}_{\text{CoT}}$. Therefore, the entropy over the predictive distribution (Eq. 5) can be expressed as:

$$(E_t \mid Q, \mathbf{s}_t, \mathbf{I}_{\text{CoT}}) = (E_t \mid \mathbf{x}_t) = -\sum_{e} p_\theta(e \mid \mathbf{x}_t) \log p_\theta(e \mid \mathbf{x}_t), \quad e \in \{0,1\}. \tag{6}$$

CoT Entropy is an approximation of this entropy by marginalizing over some rationales c:

$$\begin{aligned} \text{CoTE}(\mathbf{x}_t) &= -\sum_{e} \Big(\Big[\sum_{c} p_\theta(c, e \mid \mathbf{x}_t) \Big] \log \Big[\sum_{c} p_\theta(c, e \mid \mathbf{x}_t) \Big] \Big) \\ &= -\sum_{e} \Big(\Big[\sum_{c} p_\theta(e \mid \mathbf{x}_t, c) p_\theta(c \mid \mathbf{x}_t) \Big] \log \Big[\sum_{c} p_\theta(e \mid \mathbf{x}_t, c) p_\theta(c \mid \mathbf{x}_t) \Big] \Big), \end{aligned} \tag{7}$$

where a reasoning path c is sampled given a context $\mathbf{x}_t$, followed by a verification decision e sampled conditioned on c:

$$c \sim p_\theta(- \mid \mathbf{x_t}), \quad e \sim p_\theta(- \mid c). \tag{8}$$

In practice, we follow three steps:

1. *Generation:* Given a judge-LM and a context $\mathbf{x}_t$, sample reasoning sequences c that critique the current step s_t, followed by a decision e (of whether s_t is deemed incorrect) after the CoT.
2. *Clustering:* Group the output sequences into two clusters based on whether the decision e equals 1 (deemed incorrect) or 0 (deemed correct).
3. *Entropy estimation:* Normalize token probabilities $\tau(e)$ to obtain class probabilities using Eq. 5 for each sequence in both clusters. Then, sum the class probabilities of sequences leading to the same decision e and compute the resulting entropy.

We highlight that, crucially, by conditioning the token probability of $\tau(e)$ on the reasoning c and marginalizing over these critiques, the CoT verifier naturally generates a posterior predictive distribution $p_\theta(E_t = e \mid \mathbf{x}_t) = \sum_{c} p_\theta(e \mid \mathbf{x}_t, c)\, p_\theta(c \mid \mathbf{x}_t)$ over the decisions e. This formulation treats different CoTs c as distinct justifications generated by the judge-LM backing its final verification judgment about the current step s_t, ensuring that the judgment is supported by the reasoning. Unlike methods that generate explanations after a model reaches a conclusion (Zheng et al., 2023; Wang et al., 2023), our approach integrates reasoning into the decision-making process. Furthermore, our method leverages the full (approximated) posterior predictive distribution to obtain a probabilistic measure of confidence across different outcomes, differing from self-consistency (Wang et al., 2022), which relies solely on majority voting over samples from the distribution for point estimation.

4 Uncertainty Quantification on Math Reasoning Verification

In this section, we detail the experimental setup for measuring the performance of different uncertainty quantification methods u on step-wise math reasoning verification.

4.1 Datasets

We conduct experiments on the process supervision dataset PRM800K (Lightman et al., 2023), which consists of 800k step-level correctness labels for model-generated solutions to problems from the MATH (Hendrycks et al., 2021), a challenging competition math dataset. During annotation, data labelers were presented with step-by-step solutions to MATH problems pre-generated by GPT-4 and assigned each step a label of positive, negative, or neutral. We consider the negative class $y = 1$, indicating the presence of an error, while converting all other labels to $y = 0$. Due to computational constraints, we select a subset of 150 out of 500 questions from the test split, resulting in a total of 1,152 steps for judge-LM verification. Among these annotated steps, 129 (11.2%) are labeled as containing errors ($y = 1$), leading to a class imbalance. The average number of steps per solution is 7.7.

4.2 Models

We use Qwen2-Math-72B-Instruct[1] as our judge-LM. Qwen2-Math-72B-Instruct is a specialized mathematical language model built upon the Qwen2 LLMs, achieving strong performance on math benchmarks such as GSM8K (Cobbe et al., 2021). We use the instruct version, as empirically it demonstrates a better understanding of the task of verifying mathematical solutions. Following Farquhar et al. (2024), for each step s_t we sample a single generation at low temperature (0.1) as the predicted label $\hat{e}_t$, which is used to assess the accuracy with the ground truth label y_t, and 10 generations at high temperature (1.0) producing diverse reasoning paths c and decisions e for clustering.

4.3 Baseline Methods

We compare CoT Entropy against the following baseline methods. **Naive Entropy** is the length-normalized average log token probabilities across the same number of generations as for CoT Entropy. **P(True)** (Kadavath et al., 2022) estimates uncertainty by prompting an LM to compare a main answer with 'brainstormed' alternatives and using the predicted probability that the main answer is 'True'. We use 0-shot prompting in our experiments where judge-LM self-reflects how likely its greedy verification result is compared to 10 alternatives. **SEU** (Grewal et al., 2024) is an embedding-based approach that measures uncertainty via the average pairwise cosine similarity of the embeddings of the generated responses. The intuition is that uncertain responses exhibit lower cosine similarity (greater semantic diversity) among outputs. This method relies on sentence embedding models to map semantically similar responses to nearby points in the embedding space. We use MiniLM (Wang et al., 2020) and NV-Embed (Lee et al., 2024) as embedding models for SEU. **CoT Entropy (Discrete)** is a discrete variant of our proposed CoT Entropy. Following semantic entropy (Farquhar et al., 2024), it replaces token probabilities in Eq. 7 with empirical decision class frequencies. As it does not rely on probabilities, it is a black-box method. Additionally, we include a **Random** baseline, which samples an uncertainty score between 1 and 0 for each example. For all uncertainty methods, we run experiments with 5 random seeds and report the mean and standard deviation of their performance. Note that our comparisons focus on different uncertainty estimates u_t, rather than the underlying PRM we use for the judge-LM, as uncertainty quantification is a task independent of the PRM's predictive performance.

4.4 Evaluation metrics

We evaluate uncertainty quantification methods using the following metrics, each based on an uncertainty score estimated by u_t assessed against reference labels y_t.

[1] `https://qwenlm.github.io/blog/qwen2-math/`

AUROC. Following previous works (Farquhar et al., 2024; Kossen et al., 2024), we use the area under the receiver operating characteristic curve (AUROC) for the binary event that a given verification is correct. AUROC ranges from 0 to 1, where 1 indicates a perfect classifier and 0.5 corresponds to an uninformative classifier. **AUPRC.** We also include the area under the precision-recall curve (AUPRC), where the baseline for an uninformative classifier is equal to the proportion of positive samples in the dataset, which, in our case, corresponds to the proportion of correctly verified steps.

AU-F1C and Rejection-F1. We use selective performance scores to evaluate how well a model’s uncertainty estimates can reject verification steps likely to lead to errors. While accuracy is commonly used in previous works (Farquhar et al., 2024), we opt for F1-score due to the observed class imbalance (11.2% positive labels) – ensuring a fairer evaluation – while also highlighting the importance of identifying positive cases, which indicate that an intermediate step contains an error. **Rejection-F1** measures the F1-score on the retained steps after filtering by uncertainty. We define “Rejection-F1 at *X%*” as the F1-score computed on the most confident *X%* of inputs, as determined by the uncertainty estimates. “Rejection-F1 at *100%*” yields the same performance across all methods, equivalent to an uninformative classifier. The area under the F1 curve (**AU-F1C**) serves as a summary statistic across multiple uncertainty thresholds, capturing the potential F1 improvement a user would experience when filtering out the most uncertain steps. In our experiments, AU-F1C is computed from 30% onward to avoid a low-data regime below this threshold, and Rejection-F1 is reported for thresholds from 60% to 100%.

5 Evaluation Results

We evaluate the different uncertainty quantification methods u initialized with Qwen-2-Math-72B-Instruct on a subset of PRM800K test set as detailed in §4. We show the main evaluation results in §5.1 and analyze the sources of uncertainty in step-wise verification in §5.2.

5.1 Does CoT Entropy inform confidence in verification?

We report AUROC, AUPRC, and AU-F1C with standard deviation across the five runs in table 1. Overall, CoT Entropy achieves the highest performance across the three metrics, indicating its effectiveness in distinguishing between correctly and incorrectly verified steps. We note that the baseline values for AUROC and AUPRC differ: AUROC is approximately 0.5 for an uninformative classifier that randomly assigns prediction correctness labels, while the AUPRC baseline corresponds to the proportion of correctly verified steps (0.786). For more details on verification performance, see table 2 in App. A.2.

Table 1: Comparison of different uncertainty estimation methods.

Method	**AUROC**	**AUPRC**	**AU-F1C**
Random	$0.521_{0.024}$	$0.786_{0.010}$	$0.279_{0.019}$
Naive Entropy	$0.408_{0.013}$	$0.742_{0.004}$	$0.265_{0.009}$
P(True)	$0.329_{0.015}$	$0.671_{0.007}$	$0.302_{0.007}$
SEU-Minilm	$0.661_{0.029}$	$0.862_{0.055}$	$0.304_{0.010}$
SEU-NV-Embed	$0.616_{0.011}$	$0.759_{0.009}$	$0.330_{0.010}$
CoT Entropy	$\mathbf{0.680}_{0.017}$	$\mathbf{0.885}_{0.005}$	$\mathbf{0.348}_{0.009}$

Fig. 1 presents the Rejection-F1 of various uncertainty quantification methods. The *x*-axis represents the rejection threshold, ranging from 60% to 96%, while the *y*-axis shows the F1-score on the retained examples, corresponding to the most confident *X%* of the total examples. The dashed line in the background represents the 100% threshold (1,152 steps in total), where no uncertainty method is applied. This reflects the model’s performance on the original unfiltered test set, and also serves as a reference performance equivalent to the random baseline. We note that with 60% threshold, the error bars are larger due to the lower number of steps (691) in this bin. If an uncertainty method is effective, predictions on the confident subset should achieve a higher performance score than those on the excluded subset, with Rejection-F1 improving as more inputs are rejected. This trend is evident in the plot, where Rejection-F1 increases from right to left, indicating that these methods generally produce meaningful confidence estimates for step-wise verification.

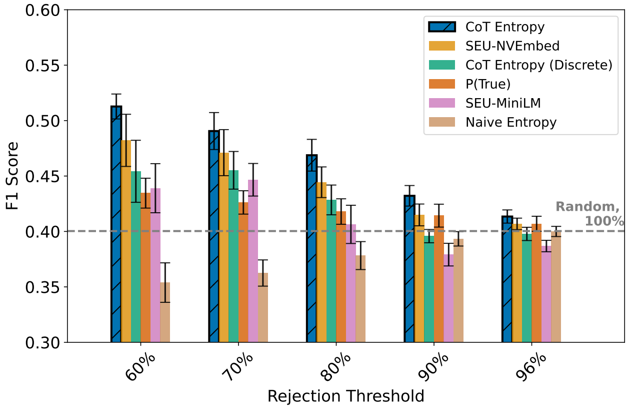

Figure 1: **Rejection-F1 for different uncertainty quantification (UQ) methods.** The bars represent the F1-Score on the retained examples, corresponding to the $X\%$ most confident examples as determined by the UQ method, at the rejection threshold on the *x*-axis. CoT Entropy outperforms leading baselines in detecting the correctness of step-wise verification for intermediate reasoning traces for solving math problems. Results are averaged over five runs.

We observe that CoT Entropy achieves the highest Rejection-F1 among all thresholds, indicating the method’s effectiveness in detecting verification errors. The naive entropy baseline, which lacks clustering, performs worse than random, highlighting the importance of structuring outputs into meaningful clusters for entropy-based methods. CoT Entropy (Discrete) underperforms CoT Entropy, suggesting that access to probabilities enhances entropy estimation. This differs with semantic entropy (Farquhar et al., 2024), where little performance gap is observed between discrete and non-discrete variants. Aside from the fixed number of clusters in our setup, we hypothesize that the difference arises from the complexity of the math verification task. In our setting, probabilities capture nuances conditioned on preceding reasoning, meaning that even when two outputs reach the same decision, their probabilities may differ based on the reasoning path taken.

Comparing the embedding-based SEU methods, we observe that NV-EMBED outperforms MINILM, indicating that a higher-quality embedding model maps outputs to a more meaningful continuous space, leading to better estimation of semantic diversity. Notably, CoT Entropy outperforms SEU, which relies on an external embedding model. This suggests that when the judge-LM produces a meaningful predictive distribution, its token probabilities alone can enable more effective uncertainty quantification without relying on external models for heuristics.

Nevertheless, reward verification on reasoning is a challenging task, and the Rejection-F1 performance indicates room for improvement. However, we have shown that the proposed CoT Entropy method effectively informs the judge-LM’s confidence in step-wise verification. Using selective verification for steps with lower uncertainty improves verification reliability for reward models, particularly in complex reasoning tasks.

5.2 HOW DO DIFFERENT UNCERTAINTY TYPES PERFORM IN THE VERIFICATION TASK?

To understand the relevance of different sources of uncertainty present in the step-wise verification setup, we decompose the total predictive uncertainty associated with each data point into an epistemic and an aleatoric components (Malinin & Gales, 2020):

$$\underbrace{\mathcal{I}\big[E_t, \boldsymbol{c}|Q,\mathcal{D}\big]}_{\text{Epistemic Uncertainty}} = \underbrace{\mathcal{H}\big[\mathrm{P}(E_t|Q,\mathcal{D})\big]}_{\text{Predictive Uncertainty}} - \underbrace{\mathbb{E}_{\mathsf{q}(c|\mathcal{D})}\big[\mathcal{H}[\mathrm{P}(E_t|Q;c)]\big]}_{\text{Aleatoric Uncertainty}}. \tag{9}$$

Equation 9 computes the Epistemic Uncertainty of a Bayesian model as the Mutual Information between the predicted variable and the hypotheses variable (Lindley, 1956). In our case, these variables are, respectively, the step-wise verification E_t and the rationales c. Furthermore, $q(c|\mathcal{D})$ represents the posterior over rationales, which is given by our generator model conditioned on the in-context dataset $\mathcal{D}$. Empirically, we approximate this posterior over rationales with Monte-Carlo

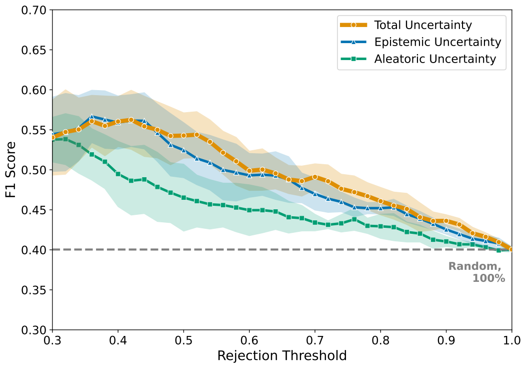

Figure 2: **Decomposition of the total predictive uncertainty**. As expected for a verification task, total uncertainty better captures the verifier's mistakes. Epistemic uncertainty performs similarly, suggesting that most errors in math reasoning verification are associated with model uncertainty rather than label noise.

sampling, which allows us to easily approximate the posterior predictive distribution and the terms in Equation 9.

We employ the three different types of uncertainty in the verification task and report the Rejection-F1 in Figure 2. First, we highlight that all uncertainty sources improve over a random baseline, showing that the uncertainty estimations over the rationales are indeed effective in identifying the questions where the verifier potentially does not know the answer. We also observe that the predictive uncertainty presents the best Rejection-F1. This is expected given the nature of a verification task: we hope to find *any* point that may lead to a prediction error, regardless of the underlying reason. This contrasts with other problem settings, such as Bayesian Active Learning, where the focus is on epistemic uncertainty as a way to improve the model's knowledge (Lindley, 1956; Gal et al., 2017; Carvalho Melo et al., 2024).

Furthermore, we observe that the epistemic uncertainty (representing the model's uncertainty or lack of knowledge) better correlates with the prediction errors than the aleatoric uncertainty (representing the uncertainty about the label-generating process – for instance, a disagreement between the labelers). In fact, epistemic uncertainty is almost as good as the total uncertainty in predicting the verifier prediction errors. Overall, this suggests an interesting insight about the problem setting: most of the identified errors are associated with the model's uncertainty – indeed, improving the capacity of LLMs for math reasoning remains an active area of research.

6 Discussion

In this work, we explored using uncertainty quantification (UQ) as a principled way to improve the reliability of step-wise verification using generative reward models on math reasoning tasks. We extend the application of UQ beyond standard QA to the domain of complex reasoning. We proposed CoT Entropy, a novel method that informs the judge-LM's confidence in verifying step-wise reasoning process in a PRM dataset, outperforming other baseline UQ methods. One limitation of our work is that the prompt for judge-LM can be further optimized. For future work, we aim to leverage the estimated uncertainty to address and potentially mitigate reward hacking during RL training while also aiding inference-time search. Additionally, we plan to fine-tune a generative RM to assess whether it can produce more accurate uncertainty estimates for verification.

References

Luckeciano Carvalho Melo, Panagiotis Tigas, Alessandro Abate, and Yarin Gal. Deep bayesian active learning for preference modeling in large language models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 118052–118085. Curran Associates, Inc.,

2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/d5e256c988bdee59a0f4d7a9bc1dd6d9-Paper-Conference.pdf.

Luís Felipe P Cattelan and Danilo Silva. How to fix a broken confidence estimator: Evaluating post-hoc methods for selective classification with deep neural networks. *arXiv preprint arXiv:2305.15508*, 2023.

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.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, 2024.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 1183–1192. JMLR.org, 2017.

Yashvir S Grewal, Edwin V Bonilla, and Thang D Bui. Improving uncertainty quantification in large language models via semantic embeddings. *arXiv preprint arXiv:2410.22685*, 2024.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

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. *arXiv preprint arXiv:2103.03874*, 2021.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Jannik Kossen, Jiatong Han, Muhammed Razzak, Lisa Schut, Shreshth Malik, and Yarin Gal. Semantic entropy probes: Robust and cheap hallucination detection in llms. *arXiv preprint arXiv:2406.15927*, 2024.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nv-embed: Improved techniques for training llms as generalist embedding models. *arXiv preprint arXiv:2405.17428*, 2024.

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

D. V. Lindley. On a Measure of the Information Provided by an Experiment. *The Annals of Mathematical Statistics*, 27(4):986 – 1005, 1956. doi: 10.1214/aoms/1177728069. URL https://doi.org/10.1214/aoms/1177728069.

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

Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. Iterative reasoning preference optimization. *arXiv preprint arXiv:2404.19733*, 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, 2024. URL https://arxiv.org/abs/2412.05563.

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.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations, 2024. URL https://arxiv.org/abs/2312.08935.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33:5776–5788, 2020.

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.

Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, et al. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization. *arXiv preprint arXiv:2306.05087*, 2023.

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

Zihuiwen Ye, Fraser Greenlee-Scott, Max Bartolo, Phil Blunsom, Jon Ander Campos, and Matthias Gallé. Improving reward models with synthetic critiques, 2024. URL https://arxiv.org/abs/2405.20850.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction, 2024. URL https://arxiv.org/abs/2408.15240.

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. *arXiv preprint arXiv:2501.07301*, 2025.

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. *arXiv preprint arXiv:2412.06559*, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623, 2023.

A Appendix

A.1 Prompt Template

We show the template we use to prompt LLMs for step-wise verification in Fig. A.1. During our experiments, we have tested various prompting templates and observed that it is crucial to use a prompt that increases the JSON parsing success rate, as it serves as an indicator of the model’s understanding of the verification task.

A.2 Verification Performance

Template for Step-wise Verification

You are a professional mathematician. Given a problem and the previously proposed steps, your task is to evaluate whether the next proposed step contains any errors in relation to the preceding steps, giving your reasoning. If there is an error, state “yes”. If there is no error, state “no”. Focus solely on the transition from the previous step to the next.
The evaluation should adhere to the following criteria:

1. Accuracy: Verify all calculations, including algebraic manipulations and numerical computations, are correct.
2. Logical Progression: Ensure the next proposed step follows logically from the previous step, applying mathematical rules, theorems, or formulas correctly, and making reasonable observations. Note: Omit this criterion if the step being evaluated is the first step, as there are no preceding steps to compare.
3. Step-by-Step Focus: Evaluate only the immediate transition from the previous step to the next proposed step. Do not mark the next step as incorrect for not performing an action that should logically occur in a future step.

Problem: {`user_problem`}
Preceding Steps: {`solution_so_far`}

Next Proposed Step to be Evaluated: {`next_step`}

Now, generate your response in the following JSON format. Give your reasoning in short and concise sentences after “reasoning”, then output your final evaluation after “has_error”.

```
{"reasoning":  "Your reasoning here.", "has_error":
"yes/no"}
```

Table 2: Comparison of verification performance with Qwen2-Math-72B-Instruct, prompted with or without CoT. Although No-CoT achieves higher accuracy, this is largely due to the model predominantly predicting a label of 0 (no error). Since errors constitute the minority class in this imbalanced dataset,the F1 score provides a more meaningful evaluation, as it better reflects the model’s ability to detect errors.

Method	**F1**	**Acc.**
No-CoT prompted	$0.352_{0.010}$	$0.838_{0.003}$
CoT prompted	$0.400_{0.003}$	$0.774_{0.002}$
Predicting all 0s	0.0	0.889
Predicting all 1s	0.197	0.112