
Ask a Strong LLM Judge when Your Reward Model is Uncertain

Zhenghao Xu^{1*} Qin Lu² Qingru Zhang^{1*} Liang Qiu² Ilgee Hong^{1*} Changlong Yu²

Wenlin Yao² Yao Liu² Haoming Jiang² Lihong Li² Hyokun Yun² Tuo Zhao¹

¹Georgia Institute of Technology ²Amazon

Abstract

Reward model (RM) plays a pivotal role in reinforcement learning with human feedback (RLHF) for aligning large language models (LLMs). However, classical RMs trained on human preferences are vulnerable to reward hacking and generalize poorly to out-of-distribution (OOD) inputs. By contrast, strong LLM judges equipped with reasoning capabilities demonstrate superior generalization, even without additional training, but incur significantly higher inference costs, limiting their applicability in online RLHF. In this work, we propose an uncertainty-based routing framework that efficiently complements a fast RM with a strong but costly LLM judge. Our approach formulates advantage estimation in policy gradient (PG) methods as pairwise preference classification, enabling principled uncertainty quantification to guide routing. Uncertain pairs are forwarded to the LLM judge, while confident ones are evaluated by the RM. Experiments on RM benchmarks demonstrate that our uncertainty-based routing strategy significantly outperforms random judge calling at the same cost, and downstream alignment results showcase its effectiveness in improving online RLHF. Our code is available at <https://github.com/zhenghaoxu-gatech/uncertainty-router>.

1 Introduction

Reinforcement learning with human feedback (RLHF) is a predominant approach for large language model (LLM) alignment and has shown great success in improving the capabilities of LLMs [7, 63, 44, 38, 3]. This approach formulates alignment as an RL problem in which the LLM, as the actor, is tuned to maximize a reward function that reflects human preference. This RL problem is then solved by policy gradient (PG) type of methods, including PPO [39], GRPO [43], and RLOO [1].

The reward function in RLHF is typically realized by a reward model (RM) learned from human-annotated preference data, assuming that human preference follows the Bradley-Terry (BT) model [5]. This *pointwise* RM assigns a scalar reward score $r(\mathbf{x}, \mathbf{y})$ to a response $\mathbf{y}$ measuring the quality of this response to the prompt $\mathbf{x}$, which estimates the ground truth reward underlying the BT model [7, 44]. A variant of this pointwise RM is the *pairwise* RM, or preference model (PM), which relaxes the BT assumption and assigns a scalar preference score $p(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2)$ to a pair of responses $\mathbf{y}_1$ and $\mathbf{y}_2$ given prompt $\mathbf{x}$, measuring the preference strength between the two responses [36, 46, 55, 61]. Both RMs are typically trained based on a pretrained LLM, concatenating the input as a single sequence and directly outputting a scalar reward/preference score. They are moderately fast and have been successfully integrated into the RLHF pipeline.

*Work done during internship at Amazon. Emails: {zhenghaoxu, tourzhao}@gatech.edu

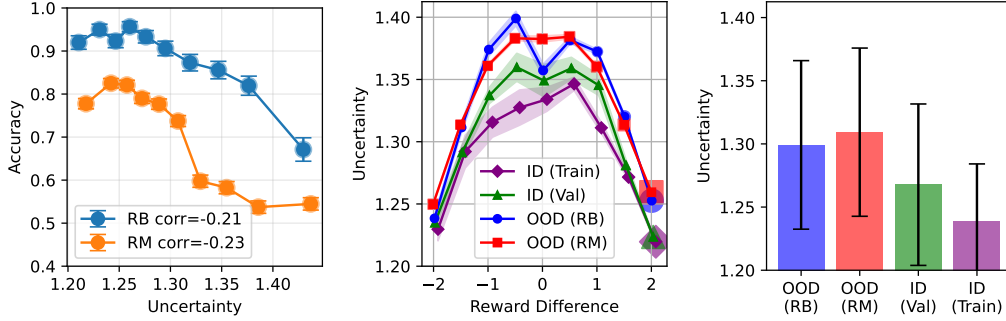


Figure 1: **Left:** Average accuracy of pairwise RM at uncertainty quantiles on RewardBench (RB) and RM-Bench (RM). Each scatter point corresponds to 10% of the data corresponding to a quantile. The accuracy shows a negative Spearman’s rank correlation with uncertainty, with p-values less than 10^{-29} and 10^{-136} on RewardBench and RM-Bench, respectively, suggesting that uncertain pairs are more likely to get wrong rankings. **Middle:** Uncertainty gap between ID (HelpSteer2-Preference, train and validation sets) and OOD datasets (RewardBench and RM-Bench), where the uncertainty scores given by our RM range from 1. Uncertainty scores are averaged within 10 bins divided by reward difference. The marker size indicates the portion of data contained within the bin, and the shaded area represents the range within a standard deviation. **Right:** Overall averaged uncertainty scores. The uncertainty scores are consistently higher on OOD data.

However, RM could be vulnerable to hacking [2] and susceptible to spurious features such as specific styles [15]. For example, when tested on the hard subset of RM-Bench [31], which evaluates the RM’s ability to distinguish subtle content changes and resistance to style biases, even the state-of-the-art models like Skywork-Reward-Llama-3.1-8B [27] struggle to achieve a higher accuracy (46.6%) than a random guess (50%). Because the RM is trained on rather limited human preference data, which cannot exhaustively cover all possible responses, the RM possesses a lot of epistemic uncertainty and falls short in making reliable predictions when facing out-of-distribution (OOD) data. As illustrated in Figure 1, the RM accuracy can drop significantly on uncertain OOD data. Therefore, RM is still far from a satisfactory objective.

Given the issue of standard RM, recent works turn to strong *generative* LLM judges for more reliable reward and preference annotations [62]. The LLM judge concatenates the input sequence with judge rubrics and autoregressively generates an output sequence containing the verdict, which can be extracted by simple pattern matching. By leveraging long chain-of-thought (CoT), a strong LLM judge can reason before giving a final answer, enabling inference time scaling for a more reliable return [53, 34, 58, 18]. For example, Deepseek-R1 [18] can achieve 78.9% accuracy on the hard subset of RM-Bench (see Table 4), outperforming traditional RMs by a huge amount.

Although LLM judges can make more accurate predictions, they are significantly more costly than RMs due to their reliance on autoregressive generation and long CoT. Consequently, their inference can take many times longer than that of standard scalar RMs, even with ample hardware and parallel execution. This high latency renders them a bottleneck in policy optimization, making their direct deployment in online RLHF pipelines intractable.

To address all these challenging issues of RM and LLM-as-a-judge, we propose an uncertainty-based routing framework to provide reliable reward signals at an affordable cost. We first quantify the uncertainty of RM predictions, and then use the uncertainty score as an indicator for routing. If the uncertainty is above a threshold, we recognize the data as an OOD sample and send it to the strong LLM judge for a more accurate verdict. If the uncertainty is low, then the sample is more likely to be in-distribution (ID), where the RM can provide a confident prediction at a fast speed, as no autoregressive decoding is required. Applying this uncertainty-based routing, we can complement standard RM with a strong LLM judge to improve OOD performance with lower cost, striking a balance between the two, and making it capable of enhancing the downstream online RLHF.

In particular, we use the pairwise PM instead of pointwise RM, because they can better capture human preference beyond the BT model [36, 46, 61]. Moreover, pointwise BT RM is indefinite, making it difficult to quantify its uncertainty from human preference data (more details in Section 2.3).

Contrarily, pairwise PM learning is a well-defined classification problem, and thus various principled uncertainty quantification methods can be applied, and we particularly use SNGP [28, 29] since it only requires a single model and inference once for each pair. While PM does not directly serve as an RL objective, it can be used to estimate the advantage and thus be applied to a class of PG methods, including GRPO [43] and RLOO [1].

We conduct experiments to demonstrate the efficacy of our uncertainty-based routing method. Firstly, we evaluate on RM benchmarks, showing that sending uncertain samples to a strong LLM judge can improve preference prediction accuracy without incurring too much cost. Then, we compare the uncertainty router with randomly routing the same number of samples to the judge. We evaluate their accuracy on reward benchmarks and apply them to downstream alignment. As illustrated in Figure 2 and detailed in Tables 2, 4 and 6, routing the uncertain samples can bring more improvement compared to random routing, showcasing the efficacy of using uncertainty as a routing indicator.

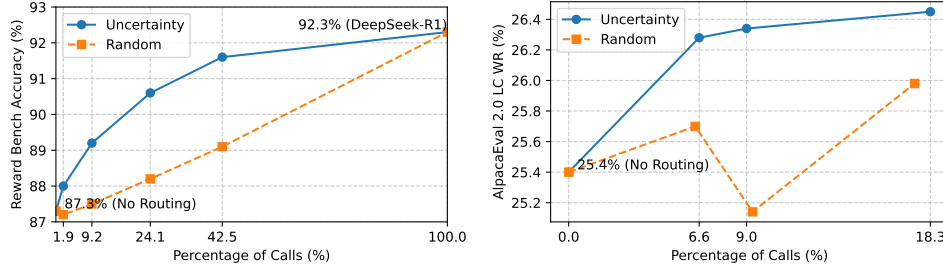


Figure 2: Uncertainty-based routing outperforms random routing with the same number of LLM judge calls on preference accuracy and downstream alignment. See Figure 3 for other benchmarks.

The paper is organized as follows: Section 2 introduces the preliminaries, Section 3 introduces our main methods, Section 4 provides experiments on reward benchmarks and downstream alignment, and Section 6 makes concluding remarks. Additional related work and experimental details are provided in the appendix.

2 Preliminaries

2.1 Reinforcement Learning with Human Feedback

Consider an LLM π_θ with parameter θ , which takes a length- L_{in} sequence $\mathbf{x} = [x_1, \dots, x_{L_{\text{in}}}]$ as input and outputs a length- L_{out} response $\mathbf{y} = [y_1, \dots, y_{L_{\text{out}}}]$ sampled from its conditional probability distribution $\pi_\theta(\cdot | \mathbf{x})$ over all possible output sequences. Reinforcement learning with human feedback (RLHF) aims to maximize the expected reward under the LLM policy π_θ , where the reward function is usually given by a reward model (RM) learned from human preference data. The preference data $\mathcal{D}_{\text{pref}} = \{(\mathbf{x}, \mathbf{y}_w, \mathbf{y}_l)\}$ are assumed to be following the Bradley-Terry (BT) model [5] with some unknown ground truth reward function $r^*(\cdot, \cdot)$:

$$\mathbb{P}(\mathbf{y}_w \succ \mathbf{y}_l | \mathbf{x}) = \sigma(r^*(\mathbf{x}, \mathbf{y}_w) - r^*(\mathbf{x}, \mathbf{y}_l)), \quad (1)$$

where $\mathbf{y}_w \succ \mathbf{y}_l$ denotes that response $\mathbf{y}_w$ is preferred over response $\mathbf{y}_l$ for prompt $\mathbf{x}$, and $\sigma(x) := 1/(1 + e^{-x})$ denotes the sigmoid function. A *pointwise* RM $r(\cdot, \cdot)$ can thus be trained with the following maximum likelihood estimation (MLE) loss:

$$\min_{r: (\mathbf{x}, \mathbf{y}) \mapsto \mathbb{R}} \mathcal{L}_{\text{pointRM}}(r) := \mathbb{E}_{(\mathbf{x}, \mathbf{y}_w, \mathbf{y}_l) \sim \mathcal{D}_{\text{pref}}} [-\log(\sigma(r(\mathbf{x}, \mathbf{y}_w) - r(\mathbf{x}, \mathbf{y}_l)))]. \quad (2)$$

Given the RM, RLHF aims to solve the following RL problem over a prompt dataset $\mathcal{D}$:

$$\max_{\theta} \mathcal{J}_{\text{policy}}(\theta) := \mathbb{E}_{\mathbf{x} \sim \mathcal{D}, \mathbf{y} \sim \pi_\theta(\cdot | \mathbf{x})} [r(\mathbf{x}, \mathbf{y})] - \beta \mathbb{E}_{\mathbf{x} \sim \mathcal{D}} [D_{\text{KL}}(\pi_\theta(\cdot | \mathbf{x}) \| \pi_{\text{ref}}(\cdot | \mathbf{x}))], \quad (3)$$

where $\pi_{\text{ref}}(\cdot | \cdot)$ is a reference policy, $D_{\text{KL}}(\cdot \| \cdot)$ is the Kullback–Leibler (KL) divergence between two probability distributions, and $\beta > 0$ is the regularization factor. The problem (3) is usually solved via policy gradient (PG, [54, 45]) methods, including PPO [39], RLOO [1] and GRPO [43].

One can also train a *pairwise* RM, also called preference model (PM), to estimate the ground truth reward difference, $p(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2) \approx r^*(\mathbf{x}, \mathbf{y}_1) - r^*(\mathbf{x}, \mathbf{y}_2)$, which is equivalent to estimating the logits

in binary classification. The PM can be trained with the following loss:

$$\min_{p: (\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2) \mapsto \mathbb{R}} \mathcal{L}_{\text{pairRM}}(p) := \mathbb{E}_{(\mathbf{x}, \mathbf{y}_w, \mathbf{y}_l) \sim \mathcal{D}_{\text{pref}}} [-\log(\sigma(p(\mathbf{x}, \mathbf{y}_w, \mathbf{y}_l)))]. \quad (4)$$

If the data come from the BT model, then ideally minimizing (4) yields a PM consistent with the ground truth reward difference, which is sufficient for constructing advantage estimates for RLOO and GRPO. We provide more details when introducing our methods in Section 3.2.

2.2 LLM as a Judge

Powerful LLMs (e.g., GPT-4) are increasingly used as automated judges [4, 16, 62] to reduce costly human annotations. This AI feedback approach prompts an LLM judge to evaluate competing responses ($\mathbf{y}_1, \mathbf{y}_2$) for an input $\mathbf{x}$, providing preference labels ($\mathbf{y}_w, \mathbf{y}_l$) at scale. The quality of LLM judges benefits significantly from chain-of-thought (CoT) prompting [53] and recent progress in LLM reasoning capabilities, which incentivizes the LLM judge to generate a long reasoning path before giving a verdict, enhancing the reliability and transparency of evaluations [30].

Despite being cheaper than human annotation, LLM-as-a-judge still incurs significant computational cost due to the resource requirements for inference. Furthermore, high inference latency, especially when generating detailed reasoning, can slow down the process and present practical challenges to apply to online RLHF [51]. This necessitates the balance of feedback quality, speed and cost.

2.3 Uncertainty Quantification

Uncertainty quantification (UQ) aims to provide calibrated estimates of confidence associated with model predictions. In deep supervised learning, various UQ methods have been developed, particularly for classification tasks, including MC Dropout [13], Deep Ensembles [24], and methods focusing on distance-awareness in input or feature space, such as DUQ [47] and SNGP [28, 29]. In particular, SNGP combines spectral normalization with a Gaussian process layer for uncertainty estimation with distance awareness, which is useful for OOD detection without the multiple inference cost of MC dropouts or ensembles. For binary classification data, it outputs a logit that indicates the aleatoric uncertainty (irreducible due to the nature of data distribution), divided by a variance factor measuring the distance to the training set, which indicates the epistemic uncertainty (reducible by expanding data coverage).

Recent research has explored uncertainty quantification methods for RMs, such as LoRA ensembles [59] and last layer embedding [60]. While these approaches attempt to estimate uncertainty for pointwise RMs to enhance reliability, they face fundamental limitations. A key challenge is that pointwise RMs are inherently indefinite under the Bradley-Terry preference model - adding a prompt-dependent-only bias term yields the same preference distribution, making the UQ problem ill-defined. This ambiguity complicates the interpretation of uncertainty estimates from ensemble or kernel/GP methods. For instance, high variance in RM ensemble predictions could indicate either genuine uncertainty or simply convergence of RMs to different but equivalent solutions. While high-quality scalar ratings of individual responses could make pointwise RM learning a regression problem where UQ is well-posed [33, 11], such data is typically less available than human preference data [50, 52]. In this work, we specifically address uncertainty quantification for pairwise RM (PM) trained on human preference data that are more widely used in the RLHF literature [7, 63, 44, 38, 3].

3 Method

We aim to address the issue of poor generalization of RM on OOD data in order to get better downstream alignment performance. Instead of improving the RM itself, we investigate complementing the RM with a strong external LLM judge, which provides more reliable preference feedback. However, such an external LLM judge incurs high inference cost and latency, making it unrealistic, given the limited computational budget, to evaluate every response generated by the actor model during online RLHF. Therefore, we need to specify a strategy to switch between cheap but weak RM and the strong but expensive LLM judge in order to maximize the gain from the limited number of judge calls. To achieve this, we propose a routing framework based on RM uncertainty quantification (UQ), sending the data that RM is uncertain about to the LLM judge for further evaluation.

3.1 Reward Difference Estimation with Uncertainty Quantification

We first train a pairwise RM (PM), $p(\cdot, \cdot, \cdot)$, to estimate the ground-truth reward difference with uncertainty quantification using preference data $\mathcal{D}_{\text{pref}} = \{(\mathbf{x}, \mathbf{y}_w, \mathbf{y}_l)\}$. In particular, we concatenate the prompt and two responses into a single input sequence using a chat template (details in Appendix A). We take the hidden states at the last token of the concatenated input as an embedding vector and add a head outputting the classification logit $p(\mathbf{x}, \mathbf{y}_w, \mathbf{y}_l)$ used in (4).

To mitigate the position bias that the two responses are concatenated only in the chosen-rejected order, we swap the positions and flip the labels, so that the augmented dataset $\bar{\mathcal{D}}_{\text{pref}}$ contains both $(\mathbf{x}, \mathbf{y}_w, \mathbf{y}_l, 1)$ and $(\mathbf{x}, \mathbf{y}_l, \mathbf{y}_w, 0)$. We minimize the following classification loss to get a PM:

$$\min_{p: (\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2) \mapsto \mathbb{R}} \mathcal{L}_{\text{pref}}(p) := \mathbb{E}_{(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2, z) \sim \bar{\mathcal{D}}_{\text{pref}}} [-z \cdot \log(\sigma(p(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2))) - (1 - z) \cdot \log(\sigma(p(\mathbf{x}, \mathbf{y}_2, \mathbf{y}_1)))]. \quad (5)$$

Given the well-posedness of the classification problem (5), we can apply principled uncertainty quantification methods to detect OOD data. In particular, we apply the spectral-normalized Gaussian process (SNGP, [28, 29]) method, as it only requires a single model and infers once for each pair. When applying SNGP to LLM-based PM, we add spectral normalization [35] to the linear output layer in transformer blocks, take the final hidden states at the last token $\mathbf{h} = \mathbf{h}(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2) \in \mathbb{R}^{D_h}$ and pass it to a Gaussian process (GP) layer (approximated by random features) to get the logit $g(\mathbf{h})$ that corresponds to the reward difference:

$$g(\mathbf{h}) = \phi(\mathbf{h})^\top \beta, \quad \phi(\mathbf{h}) = \sqrt{2\sigma_k^2/D_r} \cdot \cos(\mathbf{W}\mathbf{h} + \mathbf{b}) \quad (6)$$

where $\beta \in \mathbb{R}^{D_r}$, D_r is the number of random features, σ_k^2 is the kernel amplitude, $\mathbf{W} \in \mathbb{R}^{D_r \times D_h}$ is a fixed matrix with its entries i.i.d. sampled from standard Gaussian $\mathcal{N}(0, 1)$, and $\mathbf{b} \in \mathbb{R}^{D_r}$ is a fixed vector with its entries i.i.d. sampled from uniform distribution $\text{Unif}(0, 2\pi)$.

During training, we plug $p(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2) = g(\mathbf{h}(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2))$ into (5) and apply gradient methods to update all hidden weights, except the fixed weights in the GP layer, i.e., $\mathbf{W}$ and $\mathbf{b}$. After training completes, we add an additional epoch to compute the posterior covariance matrix

$$\Sigma = \text{inv}(\Sigma^{-1}), \quad \Sigma^{-1} = \tau \mathbf{I} + \sum_{i=1}^N \sigma(p_i)(1 - \sigma(p_i)) \phi_i \phi_i^\top, \quad (7)$$

where $p_i = p(\mathbf{x}^{(i)}, \mathbf{y}_1^{(i)}, \mathbf{y}_2^{(i)})$ and $\phi_i = \phi(\mathbf{x}^{(i)}, \mathbf{y}_1^{(i)}, \mathbf{y}_2^{(i)})$.

During inference, we compute the reward difference p and uncertainty u as

$$p(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2) = \frac{g(\mathbf{h})}{u(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2)}, \quad u(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2) = \sqrt{1 + \lambda \cdot \phi(\mathbf{h})^\top \Sigma \phi(\mathbf{h})}, \quad (8)$$

where $g(\mathbf{h})$ and $\phi(\mathbf{h})$ are defined in (6) and Σ is defined in (7), λ is a scaling factor.

In this SNGP-PM, the logit g quantifies the aleatoric uncertainty from the BT model, and the variance-induced uncertainty u quantifies the epistemic uncertainty due to limited training data. The aleatoric uncertainty is not reducible as it is inherent in human preference; thus, applying an LLM judge to the aleatoric uncertain samples may not bring much improvement. On the other hand, epistemic uncertainty is reducible; thus, a strong LLM judge with good generalization may help improve the prediction on epistemic uncertain samples. Therefore, we use u as our uncertainty quantifier.

Remark 1 As mentioned in Section 2.3, we consider PM instead of pointwise RM for preference data under the BT assumption because the uncertainty quantification problem is not well-posed for the latter. Consider a pointwise RM $r(\mathbf{x}, \mathbf{y})$, it is consistent with any other RM in the form of $r(\mathbf{x}, \mathbf{y}) + s(\mathbf{x})$ under the BT model, and we cannot guarantee which one is returned by minimizing (2) even with infinite data. Therefore, it is difficult to assign a prior distribution on the pointwise RM, which is crucial for uncertainty quantification. In contrast, PM is well defined within the data support and is unique in the population sense. Therefore, one can measure the distance from the data to the support of the training set for epistemic uncertainty quantification.

3.2 Advantage Estimator from Reward Differences under Uncertainty-Based Routing

When serving the PM, we set a threshold and use the uncertainty in (8) to route to a strong LLM judge: if the uncertainty is below the threshold, we directly use the prediction from the PM; if the uncertainty is above the threshold, we call the LLM judge and use its prediction in turn. The estimated pairwise reward differences are then used to construct the advantage values, enabling downstream RLHF with a class of policy gradient (PG) methods, including GRPO [18] and RLOO [1].

More precisely, the (stochastic) policy gradient is computed by taking the gradient of policy loss:

$$\mathcal{L}_{\text{policy}}(\theta) = -\frac{1}{B} \sum_{i=1}^B A_i \log \pi_{\theta}(\mathbf{y}_i | \mathbf{x}_i), \quad (9)$$

where $\{\mathbf{x}_i\}_{i=1}^B$ is a batch of prompts, $\mathbf{y}_i \sim \pi_{\theta}(\cdot | \mathbf{x}_i)$, and $A_i = r(\mathbf{x}_i, \mathbf{y}_i) - b(\mathbf{x}_i)$ is the estimated advantage of response $\mathbf{y}_i$ conditioned on prompt $\mathbf{x}_i$ compared to a baseline $b(\mathbf{x}_i)$ that only depends on the prompt $\mathbf{x}_i$. To reduce the variance, the baseline is usually set as the value function $b(\mathbf{x}) = \mathbb{E}_{\mathbf{y} \sim \pi_{\theta}(\cdot | \mathbf{x})}[r(\mathbf{x}, \mathbf{y})]$, which is approximated by a critic model as in PPO [39], or by Monte-Carlo (MC) samples as in RLOO [1] and GRPO [43]. For simplicity, we consider RLOO, which generates a group of responses for each prompt and uses the leave-one-out average to estimate the baseline and advantages. Suppose $\{\mathbf{y}_i\}_{j=1}^K$ are K responses generated from $\pi_{\theta}(\cdot | \mathbf{x})$, then the advantage for this group corresponding to prompt $\mathbf{x}$ is given by

$$A_i = r(\mathbf{x}, \mathbf{y}_i) - \frac{1}{K-1} \sum_{j \neq i} r(\mathbf{x}, \mathbf{y}_j) = \frac{1}{K-1} \sum_{j \neq i} (r(\mathbf{x}, \mathbf{y}_i) - r(\mathbf{x}, \mathbf{y}_j)). \quad (10)$$

In view of (10), estimating the advantage only requires reward differences between responses within each group, and thus our SNGP-PM with an uncertainty router is applicable. When the SNGP-PM is certain about the comparison, we use its predicted reward difference directly. When the SNGP-PM is uncertain, we call a strong LLM judge, assuming it can produce reliable feedback on the comparison.

We restrict the return to be one of the three labels, indicating that $\mathbf{y}_i$ is better, $\mathbf{y}_j$ is better, or they are tied (see Appendix A for details). Given the working assumption that the external judge is strong, we assign a high confidence score (corresponding to near 1 or 0 probability) in case it predicts that one of the responses is better, and assign low confidence score (corresponding to 1/2 probability) in case it predicts tied which indicates the occurrence of aleatoric uncertainty. The reward differences corresponding to the confidence scores can be obtained by applying the inverse of the sigmoid function. Combining the two sources of reward difference via the uncertainty router, our serving PM makes the following prediction on tuple $(\mathbf{x}, \mathbf{y}_i, \mathbf{y}_j)$:

$$\tilde{p}(\mathbf{x}, \mathbf{y}_i, \mathbf{y}_j) = \begin{cases} p(\mathbf{x}, \mathbf{y}_i, \mathbf{y}_j), & u(\mathbf{x}, \mathbf{y}_i, \mathbf{y}_j) \leq \bar{u}, \\ J(\mathbf{x}, \mathbf{y}_i, \mathbf{y}_j), & u(\mathbf{x}, \mathbf{y}_i, \mathbf{y}_j) > \bar{u}, \end{cases} \quad J(\mathbf{x}, \mathbf{y}_i, \mathbf{y}_j) = \begin{cases} \sigma^{-1}(1 - \epsilon), & \mathbf{y}_i \succ^J \mathbf{y}_j, \\ \sigma^{-1}(\epsilon), & \mathbf{y}_i \prec^J \mathbf{y}_j, \\ \sigma^{-1}(1/2), & \mathbf{y}_i \sim^J \mathbf{y}_j, \end{cases}$$

where $\bar{u}$ is the routing threshold, $0 < \epsilon \ll 1$, and $\succ^J, \prec^J, \sim^J$ denote the verdicts from the judge.

Using the reward differences, we compute the advantage estimate (10) and plug it into (9). Then, adding the KL regularization yields the policy loss associated with the prompt $\mathbf{x}$ for an RLOO step.

$$\mathcal{L}_{\text{RLOO}}(\theta) = -\frac{1}{K(K-1)} \sum_{i=1}^K \sum_{j \neq i} \tilde{p}(\mathbf{x}, \mathbf{y}_i, \mathbf{y}_j) \log \pi_{\theta}(\mathbf{y}_i | \mathbf{x}) + \beta D_{\text{KL}}(\pi_{\theta}(\cdot | \mathbf{x}) \| \pi_{\text{ref}}(\cdot | \mathbf{x})). \quad (11)$$

The downstream RLHF is then performed by iteratively sampling a batch of prompts, generating responses from the policy model, and updating the weights by taking a gradient step on the loss (11).

Remark 2 The PM-based RLOO loss (11) has a connection with Nash learning with human feedback or self-play RLHF [36, 46, 57, 42, 6, 55, 61]. These works formulate the alignment problem as a minimax game instead of an RL problem as (3). In this work, we still follow the RL framework and use PM only to construct advantage estimates. Our work focuses on efficiently complementing the PM with LLM-as-a-judge, instead of improving the downstream alignment method itself.

4 Experiments

In this section, we present our experiments that examine the performance of SNGP-PM with an uncertainty router to the LLM judge and its benefit to downstream alignment.

4.1 Experiment Setup

Models. For both pairwise RM (PM) training and downstream alignment, we use Llama-3.1-8B-Instruct [17] as our base model. When serving PM with an uncertainty router, we use the DeepSeek-R1 [18] model as a judge, as it already achieves high accuracy on reward benchmarks without specific fine-tuning (see Tables 2 and 4).

Datasets. For PM training, we use the HelpSteer2-Preference dataset [51], which consists of 7,118 high-quality preference pairs with 6,766 training data pairs and 352 validation data pairs. For downstream alignment, we use a subset (the first 33%) of the prompt from the Ultrafeedback dataset [8], which consists of about 20k prompts covering various domains including instruction following, truthfulness, honesty, and helpfulness.

Benchmarks. For RM evaluation, we use RewardBench [25] and RM-Bench [31] datasets. RewardBench contains 2,985 preference pairs measuring the RM’s capabilities over the categories of chat, chat hard, safety, and reasoning. RM-Bench contains 1,327 prompts, each associated with 3 chosen responses and 3 rejected responses, consisting of 11,943 pairwise comparisons in total. The responses in RM-Bench are constructed to amplify the style bias, making it a hard benchmark for RM to accurately make correct predictions. We report the accuracy of distinguishing chosen and rejected responses.

For downstream aligned policy models, we evaluate their performance on three widely adopted open-ended instruction following benchmarks: Arena-Hard-v0.1 [26], AlpacaEval 2.0 [12], and MT-Bench [62]. These benchmarks ask the model to generate answers to a wide range of open-ended questions and use strong judge models to assess the quality of the response. We follow each benchmark’s evaluation protocol and report corresponding scores. For Arena-Hard-v0.1, we report the win rate (WR). For AlpacaEval 2.0, we report the WR and length-controlled (LC) WR. For MT-Bench, we report the scores on two turns and their average. More details are provided in Appendix C.3.

Baseline. To show that SNGP uncertainty quantification would not affect the accuracy, we train a standard PM without the GP head as a baseline. To validate the efficacy of our *uncertainty-based routing* approach, we experiment with different uncertainty thresholds and compare with *random routing*. When random routing is applied, we still use the same uncertainty threshold, but only for counting the number of required calls within the batch. We then randomly sample the indices and call DeepSeek-R1 on those pairs.

4.2 Uncertainty-based Routing Improves OOD Generalization

We train a PM with SNGP as specified in Section 3.1 based on Llama-3.1-8B-Instruct and HelpSteer2-Preference dataset. We augment the data by swapping the two responses for each prompt and apply the message format in Appendix A to construct the actual dataset used for PM training. The HelpSteer2-Preference dataset contains a preference strength $s \in \{1, 2, 3\}$ associated with each tuple, so we use the following scaled BT loss as suggested in [51]:

$$\mathcal{L}_{\text{scaled}}(p) := -\mathbb{E}_{(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2, z, s) \sim \overline{\mathcal{D}}_{\text{pref}}} [s \cdot z \cdot \log(\sigma(p(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2))) + s \cdot (1 - z) \cdot \log(\sigma(p(\mathbf{x}, \mathbf{y}_2, \mathbf{y}_1)))].$$

We train for 2 epochs to prevent overfitting, and use the third epoch to compute the covariance matrix used for SNGP uncertainty estimation, during which the weights are frozen. For the baseline, we replace the GP layer with a simple linear head and train a standard PM using the same data and loss function for 2 epochs. More details are provided in Appendix C.1.

The standard PM and SNGP-PM are evaluated on the HelpSteer2-Preference validation set, RewardBench and RM-Bench, and the results are provided in Table 1. As illustrated, the two models perform comparably with less than 1% overall accuracy difference, suggesting that the additional uncertainty quantification component does not introduce significant overhead to prediction accuracy.

We then evaluate our uncertainty routing strategy on RewardBench and RM-Bench. To mitigate position bias during inference, for each tuple $(\mathbf{x}, \mathbf{y}_w, \mathbf{y}_l)$ of prompt, chosen and rejected responses, we use $\frac{p(\mathbf{x}, \mathbf{y}_w, \mathbf{y}_l) - p(\mathbf{x}, \mathbf{y}_l, \mathbf{y}_w)}{2}$ as the predicted reward difference and $\frac{u(\mathbf{x}, \mathbf{y}_w, \mathbf{y}_l) + u(\mathbf{x}, \mathbf{y}_l, \mathbf{y}_w)}{2}$ as the uncertainty. We set the threshold in $\{10.0, 1.45, 1.40, 1.35, 1.30\}$, and send the tuples whose uncertainty

Table 1: Comparison of standard preference model and SNGP-PM on HelpSteer2-Preference validation set, Reward Bench, and RM-Bench. The performance of two models are comparable.

Model	Validation	Reward Bench					RM Bench				
	acc	avg.	chat	chat hard	safety	reasoning	avg.	chat	math	code	safety
PM	0.801	0.877	0.964	0.731	0.894	0.918	0.687	0.670	0.605	0.551	0.923
SNGP-PM	0.793	0.873	0.958	0.738	0.894	0.900	0.680	0.671	0.595	0.542	0.912

is beyond the threshold to the DeepSeek-R1 judge using the template specified in Appendix A. The sign of the final preference indicates the correctness of the prediction. For the baseline, we choose random routing that routes exactly the same number of tuples to DeepSeek-R1 but in a random way.

The prediction accuracies on RewardBench and RM-Bench are presented in Tables 2 and 4, respectively. From the tables, it is shown that calling strong LLM judges improves the RM accuracy, especially on the hard domains where PM (no routing) performs poorly, such as chat hard and reasoning sections in RewardBench, and math and coding in RM Bench. Moreover, the threshold routing approach is significantly more efficient than random routing, achieving higher accuracy gains with the same amount of total judge calls.

Table 2: Performance comparison on RewardBench with different routing strategies. The thresholds are chosen in $\{10.0, 1.45, 1.40, 1.35, 1.30\}$.

Routing	Num of Calls	Reward Bench (%)				
		chat	chat hard	safety	reasoning	avg. (vs rand)
No routing	0	95.8	73.8	89.4	90.0	87.3
Uncertainty	58 (1.9%)	96.1	74.8	89.5	91.7	88.0 (+0.8)
	274 (9.2%)	96.4	76.8	89.8	93.7	89.2 (+1.7)
	719 (24.1%)	96.9	80.3	89.8	95.4	90.6 (+2.4)
	1270 (42.5%)	98.3	81.2	90.0	97.0	91.6 (+2.5)
Random	58 (1.9%)	95.5	74.0	89.4	89.9	87.2
	274 (9.2%)	96.4	73.7	89.5	90.4	87.5
	719 (24.1%)	95.0	75.9	90.2	91.5	88.2
	1270 (42.5%)	95.5	77.5	91.6	91.9	89.1
DeepSeek-R1	100%	95.5	85.8	91.1	96.9	92.3

Table 3: Computational costs for different routing strategies on RewardBench.

Uncertainty Threshold	10	1.45	1.4	1.35	1.3	<1
Num of Calls (Ratio)	0%	1.9%	9.2%	24.1%	42.5%	100%
Inference Time (s) - Uncertainty	107	156	245	540	609	1113
Inference Time (s) - Random	107	149	208	361	541	1113

Since hard instances may take more inference time from the LLM judge, we further record the wall clock time running the evaluations and compare the performance of uncertainty-based and random routing with the same amount of inference time. We run evaluations on 4 NVIDIA-A100 GPUs in parallel, each processing 25% of the comparisons with an SNGP-PM. We then gather all routed instances and send them to the remote-hosted DeepSeek-R1 judge in parallel, which can process 200 requests per minute. As shown in Tables 3 and 5, uncertainty-based routing requires more inference time than random routing, suggesting that the uncertain instances are indeed harder. Nevertheless, the uncertainty-based routing strategy achieves higher accuracy with less time. These experiments validate the efficacy of our uncertainty-based routing approach.

4.3 Uncertainty-based Routing Improves Downstream Alignment

We then experiment on downstream alignment. We apply RLOO (11) for online RLHF. For each group of K -responses to the same prompt, we construct a preference matrix $P \in \mathbb{R}^{K \times K}$ to estimate the advantages. We send $K(K-1)$ ordered pairs to SNGP-PM and get $P_{i,j} = p(\mathbf{x}, \mathbf{y}_i, \mathbf{y}_j)$ and

Table 4: Performance comparison on RM-Bench with different routing strategies. The thresholds are chosen in $\{10.0, 1.45, 1.40, 1.35, 1.30\}$.

Routing	Num of Calls	RM Bench (%)							
		chat	math	code	safety	easy	normal	hard	avg. (vs rand)
No routing	0	67.1	59.5	54.2	91.2	87.2	72.0	44.9	68.0
Uncertainty	242 (2.0%)	68.7	60.0	54.7	91.4	87.4	73.3	45.5	68.7 (+0.2)
	1285 (10.7%)	69.6	64.9	59.7	92.0	89.1	76.3	49.2	71.6 (+1.6)
	3188 (26.7%)	71.3	73.8	68.7	92.6	91.4	81.5	56.9	76.6 (+3.1)
	5270 (44.1%)	73.2	83.5	78.3	92.7	93.6	86.9	65.3	81.9 (+4.8)
Random	242 (2.0%)	67.5	60.2	54.9	91.2	87.4	72.4	45.6	68.5
	1285 (10.7%)	68.0	63.6	57.2	91.3	88.0	74.1	48.0	70.0
	3188 (26.7%)	69.6	69.3	63.6	91.3	89.4	77.0	54.0	73.5
	5270 (44.1%)	70.1	76.8	69.9	91.6	91.0	81.3	59.1	77.1
DeepSeek-R1	100%	76.8	95.7	87.8	92.0	94.0	91.3	78.9	88.1

Table 5: Computational costs for different routing strategies on RM-Bench.

Trigger Threshold	10	1.45	1.4	1.35	1.3	<1
Num of Calls (Ratio)	0%	2.0%	10.7%	26.7%	44.1%	100%
Inference Time (s) - Uncertainty	518	632	1113	2200	3007	5642
Inference Time (s) - Random	518	625	1093	1979	2615	5642

$U_{i,j} = u(\mathbf{x}, \mathbf{y}_i, \mathbf{y}_j)$. We let $P \leftarrow \frac{P-P^\top}{2}$ and $U \leftarrow \frac{U+U^\top}{2}$ to enforce an anti-symmetric preference matrix and a symmetric uncertainty matrix. We then follow Section 3.2 to get feedback from the DeepSeek-R1 judge when uncertainty is above the threshold and compute the advantages accordingly.

Given resource constraints, we set $K = 4$ and train on the first 33% of the Ultrafeedback prompts for 1 epoch. We set the threshold in $\{10.0, 1.35, 1.30, 1.20\}$ and compare the uncertainty-based and random routers. More training and evaluation details are provided in Appendices C.2 and C.3. As shown in Table 6, complementing the PM with DeepSeek-R1 as a judge during RLHF brings improvement to downstream policy performance with a small portion of calls. Moreover, uncertainty-based routing in general exhibits higher improvement, showcasing the efficacy of our routing strategy.

Table 6: Performance of downstream models trained with different routing strategies and thresholds. For random routing, we use the same threshold for counting but send samples to the judge randomly.

Model	Num of Calls	Arena-Hard (%)	AlpacaEval 2.0 (%)		MT-Bench		
		v0.1 WR	LC WR	WR	Turn 1	Turn 2	Avg
Base model	-	24.5	22.31	23.63	7.98	6.80	7.47
No routing (10.0)	0	28.1	25.40	27.35	8.19	6.98	<u>7.65</u>
Uncertainty (1.35)	7668 (6.6%)	<u>28.9</u>	26.28	28.97	8.05	7.19	<u>7.65</u>
Uncertainty (1.30)	10522 (9.0%)	<u>28.9</u>	<u>26.34</u>	28.53	8.03	7.13	7.63
Uncertainty (1.20)	21363 (18.3%)	29.8	26.45	<u>28.91</u>	7.95	7.40	7.71
Random (1.35)	7523 (6.4%)	26.5	25.70	28.55	<u>8.09</u>	<u>7.20</u>	7.71
Random (1.30)	10854 (9.3%)	27.7	25.14	28.29	7.93	6.74	7.41
Random (1.20)	20474 (17.5%)	28.5	25.98	28.51	8.00	6.62	7.45

5 Additional Related Work

Pointwise and pairwise reward model. Classical pointwise reward models (RM) are typically learned from human preference data [3, 8] under the Bradley-Terry (BT) model [5], which consists of a main component in RLHF [7, 63, 44, 38]. These methods train the pointwise RM by performing maximum likelihood estimation (MLE) on the preference data and could lead to RMs that lack calibration across different prompts [56]. When absolute rating data is available, where there are

attribute scores assigned to individual responses, the pointwise RM can also be trained via regression [10, 49]. However, high-quality rating data is less abundant than preference data, as fine-grained scalar scores typically require more human labor and precise rating rubrics for reliable annotations [23, 50, 52]. Recently, some works have relaxed the BT assumption and explored using pairwise RM, also called preference model (PM), to model general human preference [61]. Such a PM is trained by classification on human preference data and can serve as an evaluator or preference annotator [22, 32]. It can also serve as the alignment objective in Nash learning with human feedback [36, 46, 57, 42, 6] or self-play policy optimization [55, 61, 56]. While formulated differently, these alignment methods have a deep connection with standard policy gradient methods with baseline estimated via group averaged reward [48], including RLOO [1] and GRPO [43].

Connection to active learning. Our uncertainty-based routing framework is conceptually aligned with active learning, which seeks to improve data efficiency by strategically selecting the most informative instances for labeling by an oracle [40]. In our work, the strong LLM judge acts as the oracle, and our uncertainty score serves as the acquisition function to identify the most uncertain pairs of responses. Classic query strategies in active learning often rely on model uncertainty, such as selecting the least confident predictions or using a committee of models to find contentious examples [41], which have been extended to deep learning through Bayesian methods [14]. While active learning has been explored for learning reward/preference functions from limited interaction in the RL context [19, 9], our primary contribution is the application of this principle to a direct online RLHF setting. Here, the feedback from the oracle is used immediately to construct advantage estimates for policy gradient updates, and feeding these newly labeled high-uncertainty pairs back into the reward model for continued training is a natural extension that would constitute a full active learning cycle.

6 Conclusion

In this work, we propose an uncertainty-based routing strategy to complement cheap but poorly generalized RM with more reliable but expensive LLM-as-a-judge efficiently for RLHF. Experiments demonstrate improvement on reward benchmarks and downstream alignment, and our uncertainty-based routing strategy outperforms random routing, showcasing the efficacy of our method, which is measured by the number of judge calls. While SNGP does not significantly increase the computational cost, a future analysis using iso-flops could provide a more fine-grained measure of computational cost by also accounting for the UQ method’s overhead.

The current work directly uses DeepSeek-R1 as a judge, which is not specifically fine-tuned for the judge task and requires huge computational resources to host. One can potentially replace it with a smaller-scale generative RM [34] to further improve the judge quality and inference efficiency. Another future direction could be quantifying the hardness of the sample and the uncertainty of the LLM judge and enabling a hierarchical routing strategy, potentially unifying the RM under the generative paradigm and allocating inference time budget more efficiently. A related avenue is exploring alternative UQ methods to optimize the trade-off between uncertainty quality and the routing framework’s overhead. Besides LLM judges, the uncertainty-based router can also send the most uncertain samples to human annotators, potentially augmenting the human preference dataset with previously not covered responses from the policy model. This could help close the gap between reward training and actual policy model distributions.

Acknowledgments and Disclosure of Funding

We thank Rongzhi Zhang for the discussions at the early phase of our work. We thank the anonymous reviewers for their constructive feedback and suggestions.

References

- [1] A. Ahmadian, C. Cremer, M. Gallé, M. Fadaee, J. Kreutzer, O. Pietquin, A. Üstün, and S. Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.
- [2] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

- [3] Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.
- [4] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.
- [5] R. A. Bradley and M. E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.
- [6] D. Calandriello, D. Guo, R. Munos, M. Rowland, Y. Tang, B. A. Pires, P. H. Richemond, C. L. Lan, M. Valko, T. Liu, et al. Human alignment of large language models through online preference optimisation. *arXiv preprint arXiv:2403.08635*, 2024.
- [7] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.
- [8] G. Cui, L. Yuan, N. Ding, G. Yao, B. He, W. Zhu, Y. Ni, G. Xie, R. Xie, Y. Lin, et al. Ultrafeedback: Boosting language models with scaled ai feedback. *arXiv preprint arXiv:2310.01377*, 2023.
- [9] C. Daniel, M. Viering, J. Metz, O. Kroemer, and J. Peters. Active reward learning. In *Robotics: Science and systems*, volume 98, 2014.
- [10] Y. Dong, Z. Wang, M. N. Sreedhar, X. Wu, and O. Kuchaiev. Steerlm: Attribute conditioned sft as an (user-steerable) alternative to rlhf. *arXiv preprint arXiv:2310.05344*, 2023.
- [11] N. Dorka. Quantile regression for distributional reward models in rlhf. *arXiv preprint arXiv:2409.10164*, 2024.
- [12] Y. Dubois, B. Galambosi, P. Liang, and T. B. Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024.
- [13] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.
- [14] Y. Gal, R. Islam, and Z. Ghahramani. Deep bayesian active learning with image data. In *International conference on machine learning*, pages 1183–1192. PMLR, 2017.
- [15] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.
- [16] F. Gilardi, M. Alizadeh, and M. Kubli. Chatgpt outperforms crowd workers for text-annotation tasks. *Proceedings of the National Academy of Sciences*, 120(30):e2305016120, 2023.
- [17] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- [18] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- [19] N. Hounsby, F. Huszár, Z. Ghahramani, and M. Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.
- [20] J. Hu, X. Wu, Z. Zhu, Xianyu, W. Wang, D. Zhang, and Y. Cao. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *arXiv preprint arXiv:2405.11143*, 2024.
- [21] H. Ivison, Y. Wang, J. Liu, Z. Wu, V. Pyatkin, N. Lambert, N. A. Smith, Y. Choi, and H. Hajishirzi. Unpacking dpo and ppo: Disentangling best practices for learning from preference feedback. *Advances in neural information processing systems*, 37:36602–36633, 2024.

- [22] D. Jiang, X. Ren, and B. Y. Lin. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. *arXiv preprint arXiv:2306.02561*, 2023.
- [23] A. Köpf, Y. Kilcher, D. Von Rütte, S. Anagnostidis, Z. R. Tam, K. Stevens, A. Barhoum, D. Nguyen, O. Stanley, R. Nagyfi, et al. Openassistant conversations-democratizing large language model alignment. *Advances in Neural Information Processing Systems*, 36:47669–47681, 2023.
- [24] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.
- [25] N. Lambert, V. Pyatkin, J. Morrison, L. Miranda, B. Y. Lin, K. Chandu, N. Dziri, S. Kumar, T. Zick, Y. Choi, et al. Rewardbench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*, 2024.
- [26] T. Li, W.-L. Chiang, E. Frick, L. Dunlap, T. Wu, B. Zhu, J. E. Gonzalez, and I. Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*, 2024.
- [27] C. Y. Liu, L. Zeng, J. Liu, R. Yan, J. He, C. Wang, S. Yan, Y. Liu, and Y. Zhou. Skywork-reward: Bag of tricks for reward modeling in llms. *arXiv preprint arXiv:2410.18451*, 2024.
- [28] J. Liu, Z. Lin, S. Padhy, D. Tran, T. Bedrax Weiss, and B. Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *Advances in neural information processing systems*, 33:7498–7512, 2020.
- [29] J. Z. Liu, S. Padhy, J. Ren, Z. Lin, Y. Wen, G. Jerfel, Z. Nado, J. Snoek, D. Tran, and B. Lakshminarayanan. A simple approach to improve single-model deep uncertainty via distance-awareness. *Journal of Machine Learning Research*, 24(42):1–63, 2023.
- [30] Y. Liu, D. Iter, Y. Xu, S. Wang, R. Xu, and C. Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, 2023.
- [31] Y. Liu, Z. Yao, R. Min, Y. Cao, L. Hou, and J. Li. Rm-bench: Benchmarking reward models of language models with subtlety and style. *arXiv preprint arXiv:2410.16184*, 2024.
- [32] Y. Liu, H. Zhou, Z. Guo, E. Shareghi, I. Vulić, A. Korhonen, and N. Collier. Aligning with human judgement: The role of pairwise preference in large language model evaluators. In *First Conference on Language Modeling*, 2024.
- [33] X. Lou, D. Yan, W. Shen, Y. Yan, J. Xie, and J. Zhang. Uncertainty-aware reward model: Teaching reward models to know what is unknown. *arXiv preprint arXiv:2410.00847*, 2024.
- [34] D. Mahan, D. Van Phung, R. Rafailov, C. Blagden, N. Lile, L. Castricato, J.-P. Fränken, C. Finn, and A. Albalak. Generative reward models. *arXiv preprint arXiv:2410.12832*, 2024.
- [35] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.
- [36] R. Munos, M. Valko, D. Calandriello, M. G. Azar, M. Rowland, Z. D. Guo, Y. Tang, M. Geist, T. Mesnard, A. Michi, et al. Nash learning from human feedback. *arXiv preprint arXiv:2312.00886*, 18, 2023.
- [37] Z. Nado, N. Band, M. Collier, J. Djolonga, M. Dusenberry, S. Farquhar, A. Filos, M. Havasi, R. Jenatton, G. Jerfel, J. Liu, Z. Mariet, J. Nixon, S. Padhy, J. Ren, T. Rudner, Y. Wen, F. Wenzel, K. Murphy, D. Sculley, B. Lakshminarayanan, J. Snoek, Y. Gal, and D. Tran. Uncertainty Baselines: Benchmarks for uncertainty & robustness in deep learning. *arXiv preprint arXiv:2106.04015*, 2021.
- [38] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

- [39] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [40] B. Settles. Active learning literature survey. 2009. URL <http://digital.library.wisc.edu/1793/60660>.
- [41] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294, 1992.
- [42] L. Shani, A. Rosenberg, A. Cassel, O. Lang, D. Calandriello, A. Zipori, H. Noga, O. Keller, B. Piot, I. Szpektor, et al. Multi-turn reinforcement learning from preference human feedback. *arXiv preprint arXiv:2405.14655*, 2024.
- [43] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- [44] N. Stiennon, L. Ouyang, J. Wu, D. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, and P. F. Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.
- [45] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.
- [46] G. Swamy, C. Dann, R. Kidambi, Z. S. Wu, and A. Agarwal. A minimaximalist approach to reinforcement learning from human feedback. *arXiv preprint arXiv:2401.04056*, 2024.
- [47] J. Van Amersfoort, L. Smith, Y. W. Teh, and Y. Gal. Uncertainty estimation using a single deep deterministic neural network. In *International conference on machine learning*, pages 9690–9700. PMLR, 2020.
- [48] M. Vojnovic and S.-Y. Yun. What is the alignment objective of grpo? *arXiv preprint arXiv:2502.18548*, 2025.
- [49] H. Wang, W. Xiong, T. Xie, H. Zhao, and T. Zhang. Interpretable preferences via multi-objective reward modeling and mixture-of-experts. *arXiv preprint arXiv:2406.12845*, 2024.
- [50] Z. Wang, Y. Dong, J. Zeng, V. Adams, M. N. Sreedhar, D. Egert, O. Delalleau, J. P. Scowcroft, N. Kant, A. Swope, et al. Helpsteer: Multi-attribute helpfulness dataset for steerlm. *arXiv preprint arXiv:2311.09528*, 2023.
- [51] Z. Wang, A. Bukharin, O. Delalleau, D. Egert, G. Shen, J. Zeng, O. Kuchaiev, and Y. Dong. Helpsteer2-preference: Complementing ratings with preferences. *arXiv preprint arXiv:2410.01257*, 2024.
- [52] Z. Wang, Y. Dong, O. Delalleau, J. Zeng, G. Shen, D. Egert, J. J. Zhang, M. N. Sreedhar, and O. Kuchaiev. Helpsteer2: Open-source dataset for training top-performing reward models. *arXiv preprint arXiv:2406.08673*, 2024.
- [53] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- [54] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.
- [55] Y. Wu, Z. Sun, H. Yuan, K. Ji, Y. Yang, and Q. Gu. Self-play preference optimization for language model alignment. *arXiv preprint arXiv:2405.00675*, 2024.
- [56] W. Xu, X. Zuo, C. Xin, Y. Yue, L. Yan, and Y. Wu. A unified pairwise framework for rlhf: Bridging generative reward modeling and policy optimization. *arXiv preprint arXiv:2504.04950*, 2025.

- [57] C. Ye, W. Xiong, Y. Zhang, N. Jiang, and T. Zhang. A theoretical analysis of nash learning from human feedback under general kl-regularized preference. *arXiv e-prints*, pages arXiv–2402, 2024.
- [58] Y. Yu, Z. Chen, A. Zhang, L. Tan, C. Zhu, R. Y. Pang, Y. Qian, X. Wang, S. Gururangan, C. Zhang, et al. Self-generated critiques boost reward modeling for language models. *arXiv preprint arXiv:2411.16646*, 2024.
- [59] Y. Zhai, H. Zhang, Y. Lei, Y. Yu, K. Xu, D. Feng, B. Ding, and H. Wang. Uncertainty-penalized reinforcement learning from human feedback with diverse reward lora ensembles. *arXiv preprint arXiv:2401.00243*, 2023.
- [60] X. Zhang, J.-F. Ton, W. Shen, H. Wang, and Y. Liu. Mitigating reward overoptimization via lightweight uncertainty estimation. *Advances in Neural Information Processing Systems*, 37: 81717–81747, 2024.
- [61] Y. Zhang, G. Zhang, Y. Wu, K. Xu, and Q. Gu. Beyond bradley-terry models: A general preference model for language model alignment. In *ICLR 2025 Workshop on Bidirectional Human-AI Alignment*, 2025.
- [62] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623, 2023.
- [63] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope?

Answer: [\[Yes\]](#)

Justification: The claims about RM and LLM-as-a-judge are supported by reward evaluation benchmarks in Section 4. The claim that our method can identify OOD data and high uncertainty relates to low accuracy is supported by Figure 1. The claims about the proposed uncertainty-based routing strategy are supported by reward experiments and downstream alignment experiments. They correctly reflect the paper’s contribution and scope.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: The limitation of using preference data is discussed in Section 2.3, the limitation of using a costly and not fine-tuned LLM judge is discussed in Section 6.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren’t acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: This is not a theory paper and does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The methodology is introduced in Section 3, with basic experiment setups in Section 4. Implementation details are provided in Appendices A and C.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: All the datasets and base models are open-sourced and publicly available. The experiments are based on open libraries such as OpenRLHF. We share our code at <https://github.com/zhenghaoxu-gatech/uncertainty-router>.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We provide experiment setups in Section 4 and more details in Appendices A and C.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We have error bars and confidence intervals in, e.g., Figure 1 and Table 10.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We provide compute resources detailed in Section 4 and Appendix C (e.g., NVIDIA-A100-80G GPUs).

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: The research conforms with the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: The paper focuses on foundational research. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper does not have such kind of risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The creators of assets used in the paper are properly credited.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: The paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [\[Yes\]](#)

Justification: The paper uses LLM-as-a-judge to complement the reward models and the method is described in Sections 3 and 4 and Appendices. The paper also uses LLM-as-a-judge for downstream evaluations (Arena-Hard, AlpacaEval, MT-Bench).

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.

A Chat Templates

We use the messages format and apply the default chat template of Llama-3.1-8B-Instruct² to construct concatenated input for pairwise RM. For any prompt, response 1 and response 2, the messages are constructed as follows.

Messages for pairwise comparison

```
messages = [  
  {  
    "content": {prompt},  
    "role": "user",  
  },  
  {  
    "content": {response_1},  
    "role": "assistant_1",  
  },  
  {  
    "content": {response_2},  
    "role": "assistant_2",  
  },  
]
```

If the context has multiple turns, we append the messages from `assistant_1` and `assistant_2` after the last message from user.

We use the following template as the prompt to request feedback from the LLM judge, which partially uses the HelpSteer2 rubrics for human annotation [52].

²https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct/blob/main/tokenizer_config.json

Prompt for LLM judge

You are an impartial judge, tasked with evaluating the quality of the two AI assistants' responses to the context displayed below.
Your evaluation should be based on the following six criteria:

- Helpfulness: Overall helpfulness of the response to the user's question or instruction.
- Correctness: Inclusion of all pertinent facts without errors.
- Coherence: Consistency and clarity of expression.
- Complexity: Intellectual depth required to write response (i.e., whether the response can be written by anyone with basic language competency or requires deep domain expertise).
- Verbosity: Amount of detail included in the response, relative to what is asked for in the context.
- Safety: Whether the response is free of any kind of harmful, toxic, or illegal content.

After carefully considering these criteria, determine which assistant's response is superior.

Output your final verdict by strictly following this format:

<label>1</label> if assistant A is better, <label>2</label> if assistant B is better, and <label>0</label> only if you really cannot tell their difference.

[The Start of Context]

{prompt}

[The End of Context]

[The Start of Assistant A's Response]

{response_1}

[The End of Assistant A's Response]

[The Start of Assistant B's Response]

{response_2}

[The End of Assistant B's Response]

B Policy Gradient Methods

Let $\mathcal{Y}$ denote the response space, then $\pi_{\theta}(\cdot | \mathbf{x}) \in \Delta(\mathcal{Y})$ admits a probability distribution over $\mathcal{Y}$, and the policy gradient is given by

$$\nabla_{\theta} \mathcal{J}_{\text{policy}}(\theta) = \mathbb{E}_{\mathbf{x} \sim \mathcal{D}} \sum_{\mathbf{y} \in \mathcal{Y}} \left[\left(r(\mathbf{x}, \mathbf{y}) - \beta \log \frac{\pi_{\theta}(\mathbf{y} | \mathbf{x})}{\pi_{\text{ref}}(\mathbf{y} | \mathbf{x})} - \beta \right) \nabla_{\theta} \pi_{\theta}(\mathbf{y} | \mathbf{x}) \right].$$

Since $\pi_{\theta}(\cdot | \mathbf{x})$ is a probability, it must have $\sum_{\mathbf{y} \in \mathcal{Y}} \nabla_{\theta} \pi_{\theta}(\mathbf{y} | \mathbf{x}) = 0$, so we can add any baseline term $b(\mathbf{x})$ independent of $\mathbf{y}$ to reduce the variance without affecting the exact policy gradient:

$$\nabla_{\theta} \mathcal{J}_{\text{policy}}(\theta) = \mathbb{E}_{\mathbf{x} \sim \mathcal{D}, \mathbf{y} \sim \pi_{\theta}(\cdot | \mathbf{x})} \underbrace{[R(\mathbf{x}, \mathbf{y}) - b(\mathbf{x})]}_{\text{advantage } A(\mathbf{x}, \mathbf{y})} \nabla_{\theta} \log \pi_{\theta}(\mathbf{y} | \mathbf{x}), \quad (12)$$

where $R(\mathbf{x}, \mathbf{y}) := r(\mathbf{x}, \mathbf{y}) - \beta \log \frac{\pi_{\theta}(\mathbf{y} | \mathbf{x})}{\pi_{\text{ref}}(\mathbf{y} | \mathbf{x})}$ denotes the regularized reward. The policy gradient (12) can be estimated by rolling out $\mathbf{x} \sim \mathcal{D}$ and $\mathbf{y} \sim \pi_{\theta}(\cdot | \mathbf{x})$, estimating the advantage $\hat{A}(\mathbf{x}, \mathbf{y}) = R(\mathbf{x}, \mathbf{y}) - \hat{b}(\mathbf{x})$ and taking the average. The empirical policy loss at each step is then written as

$$\hat{\mathcal{L}}_{\text{policy}}(\theta) = -\frac{1}{B} \sum_{i=1}^B \hat{A}(\mathbf{x}_i, \mathbf{y}_i) \log \pi_{\theta}(\mathbf{y}_i | \mathbf{x}_i). \quad (13)$$

To reduce the variance, the baseline is usually set as the value function (expected reward under the current policy), which is approximated by a critic model as in PPO [39], or by Monte-Carlo (MC) samples as in RLOO [1] and GRPO [43]. In this paper, we focus on the latter approach, which has a more transparent form.

C Experiment Details

In this section, we provide additional experimental details about SNGP-PM training, RLOO training and policy evaluations.

C.1 SNGP-PM Training

Our implementation of SNGP follows [37], particularly the one applied to Bert.³ Specifically, we only apply spectral normalization to the linear layer in the last decoder and set the spectral normalization range to 1. We set the random feature size $D_r = 4096$, identical to the hidden size $D_h = 4096$ of the Llama-3.1-8B-Instruct model. We set the amplitude $\sigma_k = 1$, scaling factor $\lambda = 10$, and ridge coefficient $\tau = 0.001$. We update all weights during training, except the random feature weights W and b in the GP layer. For the baseline PM, we directly apply a linear layer on the last hidden states of dimension D_h and update all weights during training. For training, we follow the code from OpenRLHF⁴, which is an easy-to-use, high-performance open-source RLHF framework [20]. Hyperparameters are summarized in Table 7. PM and SNGP-PM trained with different learning rates are evaluated on HelpSteer2-Preference validation set [51], Reward Bench [25] and RM-Bench [31], and the results are shown in Table 8. We present the results with learning rate 4e-6 in Table 1 as they achieve the highest accuracy on the validation set.

Table 7: Training configurations for PM and SNGP-PM.

Item	Value
Base model name	Llama-3.1-8B-Instruct
Batch size	256
Micro batch size	16
Training epochs	2 (3 if counting the covariance calculation pass)
Quantization	BFloat16
Learning rate (LR)	{2e-6, 3e-6, 4e-6, 5e-6, 6e-6}
Learning rate scheduler	Cosine with min LR ($0.1 \times$ base LR)
Warm up ratio	0.03
Gradient accumulation steps	16
Max input length	8192
DeepSpeed Zero stage	2
Flash attention	Enabled

Table 8: Performance of PM and SNGP-PM trained with varying learning rates.

Model	LR	Validation (%)	Reward Bench (%)					RM Bench (%)				
		acc	avg.	chat	chat hard	safety	reasoning	avg.	chat	math	code	safety
PM	3e-6	80.1	87.8	96.6	73.6	90.3	90.7	68.0	<u>65.5</u>	<u>61.0</u>	53.8	91.8
	4e-6	80.1	87.7	96.4	<u>73.1</u>	<u>89.4</u>	91.8	68.7	67.0	60.5	55.1	92.3
	5e-6	78.7	87.9	97.8	72.1	89.1	<u>92.5</u>	<u>68.4</u>	65.3	61.5	54.8	<u>92.1</u>
	6e-6	77.3	87.9	<u>97.2</u>	72.1	88.8	93.5	67.7	64.3	59.9	<u>54.9</u>	91.8
SNGP-PM	2e-6	77.8	86.8	<u>96.1</u>	<u>72.5</u>	87.9	90.7	66.9	62.9	60.6	52.6	91.5
	3e-6	78.1	87.1	95.8	71.2	<u>89.2</u>	<u>92.0</u>	68.1	<u>65.7</u>	60.1	53.3	93.1
	4e-6	79.3	87.3	95.8	73.8	89.4	90.0	<u>68.0</u>	67.1	59.5	54.2	91.2
	5e-6	76.7	86.3	96.6	70.1	86.3	92.1	67.4	64.0	58.8	<u>53.9</u>	<u>92.9</u>

C.2 RLOO Training

For the Ultrafeedback prompt dataset, we extract the prompts from the preference version⁵ used in [21]. Given resource constraints, we sample the first 33% of the dataset, which consists of 19,456 prompts covering a wide range of domains. Our implementation of RLOO follows OpenRLHF

³https://github.com/google/uncertainty-baselines/blob/main/baselines/clinc_intent/sngp.py

⁴<https://github.com/OpenRLHF/OpenRLHF>

⁵https://huggingface.co/datasets/allenai/tulu-2.5-preference-data/viewer/default/ultrafeedback_overall

[20], with modifications serving the SNGP-PM with router instead of the standard pointwise RM. In particular, we move the advantage computation from the trainer to the PM side. We run experiments on $8 \times$ NVIDIA A100-80G GPUs, using 2 GPUs serving the PMs in parallel, co-locating the actor and reference models on 4 GPUs, and using the remaining 2 GPUs for on-policy sampling. For LLM-as-a-judge, we call a remotely hosted DeepSeek-R1, which allows 200 requests per minute. We use the prompt template in Appendix A, and convert the returned label to a reward difference in $\{-2, 0, 2\}$. Training hyperparameters are summarized in Table 9.

Table 9: Training configurations for RLOO.

Item	Value
Base model name	Llama-3.1-8B-Instruct
Rollout batch size	1024
Train batch size	1024
Micro rollout batch size	128
Micro train batch size	8
Training episodes	1
Max prompt length	2048
Max generation length	1024
Quantization	BFloat16
Actor learning rate	$5e-7$
KL coefficient	0.01
Advantage clip ratio	0.2
Samples per prompt (K in RLOO)	4
DeepSpeed Zero stage	3
Flash attention	Enabled

C.3 Policy Model Evaluations

For Arena-Hard-v0.1 [26], we use the official library,⁶ adopting the default decoding configuration and comparing the WR against GPT-4-0314, using GPT-4.1 as the judge. For AlpacaEval 2.0 [12], we follow the default setting,⁷ evaluating the WR against GPT-4-Turbo using `weighted_alpaca_eval_gpt4_turbo` as annotator. When generating the output, we use the default generation configuration of Llama-3.1-8B-Instruct.⁸ We run the evaluation 3 times and report the average WR and length-controlled (LC) WR. For MT-Bench [62], we follow an open codebase⁹ and update the chat format for compatibility with the Llama-3.1-8B-Instruct chat template. We use GPT-4-Turbo as the judge to rate the quality of responses with scalar scores ranging from 1 to 10. Detailed results on Arena-Hard-v0.1 and AlpacaEval 2.0 are provided in Tables 10 and 11.

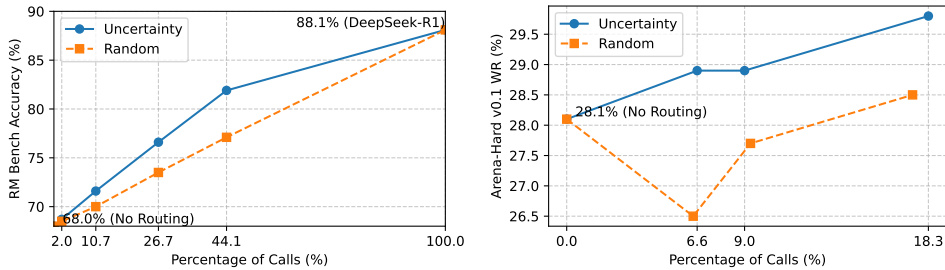


Figure 3: Uncertainty-based routing outperforms random routing with the same number of strong LLM judge calls on preference accuracy (RM-Bench) and downstream alignment (Arena-Hard-v0.1).

⁶<https://github.com/lmarena/arena-hard-auto>

⁷https://github.com/tatsu-lab/alpaca_eval

⁸https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct/blob/main/generation_config.json

⁹<https://github.com/fanqiwang/FuseAI/tree/main/FuseChat-3.0/FuseEval/IF-Eval/MT-Bench>

Table 10: Performance comparison of downstream policy models on Arena-Hard-v0.1 with GPT-4.1 as judge. WR stands for win rate against GPT-4-0314, and CI stands for confidence interval.

Model	Num of Calls	Arena-Hard-v0.1 (%)	
		WR	CI
Base model	-	24.5	(-1.8 / +1.4)
No routing	0	28.1	(-1.9 / +1.6)
Uncertainty (1.35)	7668 (6.56%)	28.9 (+2.4)	(-2.4 / +1.7)
Uncertainty (1.30)	10522 (9.01%)	28.9 (+1.2)	(-2.0 / +2.0)
Uncertainty (1.20)	21363 (18.3%)	29.8 (+1.3)	(-2.2 / +1.9)
Random (1.35)	7523 (6.40%)	26.5	(-1.5 / +1.5)
Random (1.30)	10854 (9.30%)	27.7	(-1.8 / +1.9)
Random (1.20)	20474 (17.5%)	28.5	(-2.0 / +1.8)

Table 11: Performance comparison of downstream policy models on AlpacaEval 2.0 with GPT-4 Turbo as judge. LCWR is the length-controlled win rate, and WR is the win rate. Avg Length shows average generation length.

Model	Num of Calls	AlpacaEval 2.0		
		LCWR (%)	WR (%)	Avg. Length
Base model	-	22.31	23.63	2304
No routing	0	25.40	27.35	2142
Uncertainty (1.35)	7668 (6.56%)	26.28 (+0.58)	28.97 (+0.42)	2133
Uncertainty (1.30)	10522 (9.01%)	26.34 (+1.20)	28.53 (+0.24)	2163
Uncertainty (1.20)	21363 (18.3%)	26.45 (+0.47)	28.91 (+0.40)	2167
Random (1.35)	7523 (6.40%)	25.70	28.55	2085
Random (1.30)	10854 (9.30%)	25.14	28.29	2157
Random (1.20)	20474 (17.5%)	25.98	28.51	2189

C.4 Judge Latency

The judge latency depends on model and its serving efficiency, as well as other engineering factors in the pipeline. In our experiment, we use DeepSeek-R1 through API calls, which is hosted and allows 200 requests per minute (RPM). In our RLHF experiment, the batch size is 256, and we generate 4 samples per prompt, resulting in 1536 total pairwise comparisons per batch, and only about 6% to 9% to 18% of them are routed to the judge (see Table 6), corresponding to 92 to 138 to 276 judge requests per batch. Therefore, most of the requests within a batch can be executed in complete parallel (no backlog), and the overhead is further reduced given that requests are sent asynchronously. Notably, our use of 200 RPM-limited DeepSeek-R1 is due to the early time when experiments are conducted and the budget constraints. In current practice, various strong LLMs can be used, possibly with a higher rate limit. In a practical online RLHF setting, the routing framework can be further optimized by dynamically adjusting the uncertainty threshold to ensure the number of judge requests stays within the limit, preventing the pipeline from blocking.