

Towards Reliable Alignment: Uncertainty-aware RLHF

Debangshu Banerjee¹ and Aditya Gopalan²

^{1,2} Department of Electrical and Communication Engineering, Indian Institute of Science, India

November 1, 2024

Abstract

Recent advances in aligning Large Language Models with human preferences have benefited from larger reward models and better preference data. However, most of these methodologies rely on the accuracy of the reward model. The reward models used in Reinforcement Learning with Human Feedback (RLHF) are typically learned from small datasets using stochastic optimization algorithms, making them prone to high variability. We illustrate the inconsistencies between reward models empirically on numerous open-source datasets.

We theoretically show that the fluctuation of the reward models can be detrimental to the alignment problem because the derived policies are more overfitted to the reward model and, hence, are riskier if the reward model itself is uncertain. We use concentration of measure to motivate an uncertainty-aware, conservative algorithm for policy optimization. We show that such policies are more risk-averse in the sense that they are more cautious of uncertain rewards. We theoretically prove that our proposed methodology has less risk than the vanilla method.

We corroborate our theoretical results with experiments based on designing an ensemble of reward models. We use this ensemble of reward models to align a language model using our methodology and observe that our empirical findings match our theoretical predictions.

1 Introduction

Reinforcement Learning with Human Feedback (RLHF) (Christiano et al., 2017; Ziegler et al., 2019) is an influential training approach in modern artificial intelligence research, particularly in the domain of large language models (LLMs). Notable examples include the revolutionary **ChatGPT** (OpenAI, 2023), **Claude** (Anthropic, 2023), **Gemini** (Team et al., 2023) and **LLaMA-3** (Meta, 2024). RLHF is a fine-tuning method to align the behavior of LLMs with human values and preferences. It has been instrumental in addressing challenges related to model alignment, where the goal is to ensure that an AI system adheres to specific ethical, safety, and utility guidelines defined by its

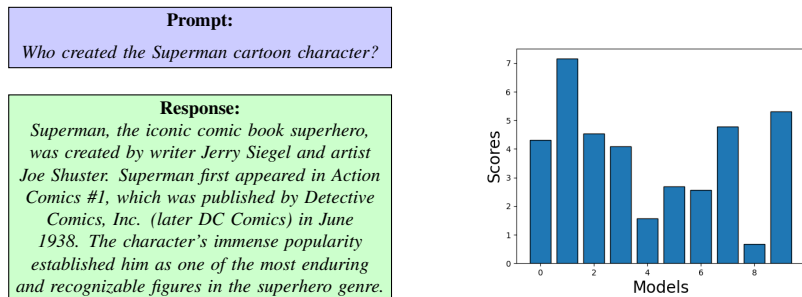


Figure 1: Reward scores assigned by 10 reward models on the same prompt-response pair. The reward models are identical in that they are trained independently on the same dataset, with the same hyperparameters and number of epochs. Despite this, we see a wide variation in the score assigned by each model.

human users. The standard reward-model RLHF framework (Ouyang et al., 2022; Bai et al., 2022b; Touvron et al., 2023) assumes a preference model based on an underlying reward model to accurately capture human preferences. The reward model is trained to predict how well a given response aligns with preferences provided by human evaluators, thus acting as a proxy for human judgment. It is a reward signal in downstream reinforcement learning to improve the LLM.

Challenges of Reward Model Reliability A critical issue in RLHF is the reliability of the learned reward model. For example, look at Figure 1, which shows the reward score assigned to the same prompt-response pair by 10 independently trained identical reward models on the same preference data. Several factors contribute to the uncertainty and potential unreliability of the reward model:

- **Limited Dataset Size:** The reward model is typically trained on a much smaller dataset than the vast corpora used to pre-train the LLM. For instance, while an LLM may be pre-trained on billions of tokens, the reward model might be trained on a few hundred thousand human-labeled prompt-response pairs. This discrepancy in the data scale can limit the generalization capability of the reward model, leading to noisy estimates of response quality.
- **Stochastic, Incomplete Optimization:** The reward model is trained using stochastic gradient descent (SGD) or variants, introducing inherent randomness into the optimization process. Using mini-batches of data means that different instances of the reward model, even when trained on the same dataset, may produce different evaluations of the same response due to the randomness in parameter updates. This stochasticity can result in high variance in the model’s predictions. Additionally, the optimization process to find a reward model is not completed – typically 1 or 2 passes over the dataset (Stiennon et al., 2020; Meta, 2024) – to avoid overfitting.

Thus, a single reward model should not be viewed as an infallible oracle for assessing response quality. Its predictions are inherently uncertain, leading to challenges when fine-tuning the LLM. Overfitting the LLM to a noisy reward model can result in

degraded performance, as the model may learn to optimize for the idiosyncrasies of the reward model rather than true human preferences.

Contributions We enumerate the contributions made in this work:

1. We provide comprehensive empirical evidence using open-source datasets to demonstrate the variability inherent in reward modeling.
2. We introduce a conservative policy optimization method incorporating uncertainty measures derived from reward model training.
3. We rigorously demonstrate, through theoretical analysis and experiments on LLMs, that our risk-aware conservative policy scheme significantly reduces the likelihood of policy degradation.

RLHF preliminaries The standard RLHF setup (Christiano et al., 2017; Ziegler et al., 2019) is described as follows. Given a prompt x , the LLM generates two responses, y^1 and y^2 . A human evaluator selects the preferred response, forming a dataset of the form $(x_i, y_i^1, y_i^2)_{i=1}^n$, where x_i is the prompt, and y_i^1, y_i^2 are model-generated responses. These pairwise comparisons encode ordinal preferences, used to train the reward model. The reward model, r_θ , assigns a scalar reward to each prompt-response pair (x, y) , reflecting its likelihood of being preferred. The Bradley-Terry model (Bradley and Terry, 1952) estimates the probability that y^1 is preferred over y^2 as: $\mathbb{P}(y^1 \text{ is preferred over } y^2) = \sigma(r_\theta(x, y^1) - r_\theta(x, y^2))$, where $\sigma(z) = \frac{1}{1+e^{-z}}$ is the logistic sigmoid function. The reward model is trained by minimizing the negative log-likelihood of human preferences: $\min_\theta \frac{1}{n} \sum_{i=1}^n -\ln \sigma(r_\theta(x_i, y_i^1) - r_\theta(x_i, y_i^2))$. This loss function seeks to adjust the parameters θ of the reward model such that the predicted rewards for preferred responses are consistently higher than those for less preferred responses, as judged by human evaluators. Using the Bradley-Terry model ensures that the reward model produces outputs that align with human feedback. Once trained, the reward model is used to fine-tune the LLM via reinforcement learning (e.g., PPO (Schulman et al., 2017)). The objective is to maximize the reward for new prompts while constraining divergence from the reference policy π_0 :

$$\max_{\pi} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi(\cdot|x)} [r_\theta(x, y)], \text{ s.t. } \text{KL}(\pi || \pi_0) \leq \varepsilon, \quad (1)$$

Solving this optimization adjusts the LLM to generate responses that align with the reward model to better reflect human preferences. However, the reward function r_θ above in Equation 1 can be inherently highly variable, as seen in Figure 1.

To illustrate the impact of uncertainty in reward models, consider a simple three-armed bandit problem. Aligning a language model can be viewed as a contextual bandit scenario where the policy assigns probabilities to each arm to maximize the expected return. In this example, the true rewards (shown in green in Figure 2) are $r_1^* < r_2^* < r_3^*$, with Arm 1 having the lowest mean reward and Arms 2 and 3 having higher rewards. However, the estimated rewards (depicted in blue as \hat{R}_1, \hat{R}_2 , and \hat{R}_3) inaccurately suggest that Arm 1 has the highest reward. If probabilities are assigned solely based on

these estimates, Arm 1 will receive the highest probability, leading to a lower true return since its actual reward is the lowest. However, when considering the uncertainty intervals (shown in red in Figure 2), it becomes evident that Arm 1’s high estimated reward comes with significant uncertainty. Arms 2 and 3 exhibit much less uncertainty, albeit having lower estimated rewards. A more conservative strategy that accounts for this uncertainty would allocate greater probabilities to Arms 2 and 3, leveraging their more reliable estimates. This example highlights the trade-off between pursuing high-risk strategies and opting for lower-reward, lower-risk approaches in policy optimization. It demonstrates the importance of incorporating uncertainty into the fine-tuning process.

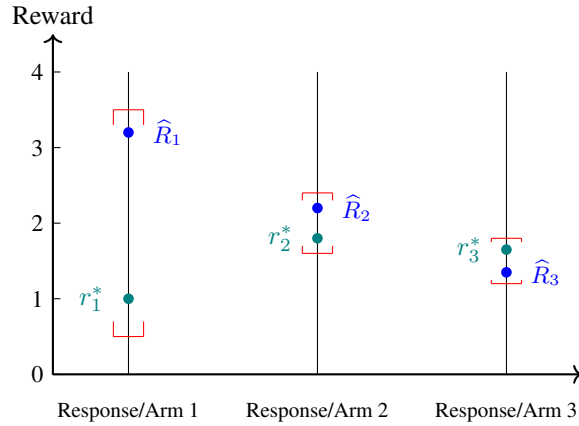


Figure 2: A 3-armed bandit problem illustrating true rewards r_1^* , r_2^* , r_3^* (green circles), estimated rewards (blue circles) \hat{R}_1 , \hat{R}_2 , \hat{R}_3 , and uncertainty intervals (red brackets). Arm 1 has the lowest true reward, whereas the highest estimate \hat{R}_1 . In contrast, arms 2 and 3 have lower reward estimates \hat{R}_2 and \hat{R}_3 , respectively. A naive policy improvement based on only the estimated rewards \hat{R}_i would increase the probability on Arm 1, leading to a lower (true) expected return. A more conservative policy improvement strategy should factor in the uncertainty of the estimate of Arm 1 and assign a lower probability to it, resulting in a higher expected return.

Related Work The pitfalls of overly relying on reward models (as proxies for actual tasks) in RLHF have been extensively documented, often referred to as *reward hacking* (Amodei et al., 2016) or *reward overoptimization* (Gao et al., 2023). For example, Shen et al. (2023) demonstrates that even large models resort to random guessing when faced with conflicting instructions and responses. Researchers have explored using reward model ensembles to address the mitigation of reward hacking (Coste et al., 2023; Eisenstein et al., 2023; Zhang et al., 2024). Leveraging conservative lower confidence bounds (LCBs) on reward to guide the training of LLMs has been investigated by Zhai et al. (2023); Xiong et al. (2024); Liang et al. (2022) and Zhang et al. (2024). Ramé et al. (2024) use a weighted average of an ensemble of reward models as a reward estimate. Methods for uncertainty quantification in deep learning using model ensembles have been studied by (Lakshminarayanan et al., 2016; Liang et al., 2022; Zhai et al.,

2023; Coste et al., 2023; Zhang et al., 2024) among others. Other approaches include Lou et al. (2024), where a reparameterization trick is used to learn uncertainties, similar to the dropout method employed by Gal and Ghahramani (2016). In this work, we utilize an ensemble of reward models to help quantify reward uncertainty. Our approach mirrors the ensemble reward modeling method of Zhang et al. (2024); however, we enhance the training efficiency by freezing the foundation layers when creating ensembles. Our problem formulation is also distinct from the LCB estimates used in previous studies, and offers a principled and practical approach to leverage uncertainty in reward models to perform reliable policy improvement.

2 Mathematical Modeling

Notations: We assume that prompts are strings denoted by x from a prompt set \mathcal{X} , and responses are strings denoted by y from a response set \mathcal{Y} . A reward model assigns a scalar value to each prompt-response pair (x, y) . We consider the learned reward model \hat{R} as a sample estimate of the true human-representative reward model r^* . Assuming \mathcal{X} and \mathcal{Y} are finite with cardinalities X and Y , respectively, both \hat{R} and r^* can be viewed as elements of \mathbb{R}^{XY} . A large language model, for our purposes, is a policy π that defines a distribution over responses \mathcal{Y} given a prompt x . We also introduce a distribution \mathcal{D} over prompts, representing their ambient frequency in nature. With a slight abuse of notation, we treat the policy π as the induced joint distribution over prompts and responses. This allows us to simplify notation by expressing the average reward $\mathbb{E}_{\substack{x \sim \mathcal{D} \\ y \sim \pi(\cdot | x)}} [\hat{R}(x, y)]$ as $\hat{R}^\top \pi$. We denote a covariance matrix by Σ , use $\|x\|_2$ to represent the Euclidean (ℓ^2) norm, and define $\|x\|_\Sigma^2$ as the quadratic form $x^\top \Sigma x$.

Noisy Reward Model We consider the true reward function r^* , which is unknown, and the learned reward model \hat{R} , which estimates r^* but is subject to noise due to finite and imperfect training data. We assume:

Assumption 2.1. For any (x, y) , the estimated reward $\hat{R}(x, y)$ is a Gaussian perturbation of $r^*(x, y)$:

$$\hat{R}(x, y) = r^*(x, y) + \mathcal{N}(0, \sigma^2(x, y)),$$

where $\mathcal{N}(0, \sigma^2(x, y))$ is a Gaussian random variable with mean zero and variance $\sigma^2(x, y)$. We assume that the estimates $\hat{R}(x, y)$ are independent across different (x, y) .

Thus, $\hat{R} \sim \mathcal{N}(r^*, \Sigma)$, where Σ is a diagonal matrix with entries $\sigma^2(x, y)$. Our goal is to optimize the policy π to maximize the expected reward estimated by \hat{R} . Let π_0 be a reference policy (e.g., from pre-training), and define $d = \pi - \pi_0$. Since $\hat{R} \sim \mathcal{N}(r^*, \Sigma)$, the scalar $\hat{R}^\top d$ is normally distributed with mean $r^{*\top} d$ and variance $d^\top \Sigma d$: $\hat{R}^\top d \sim \mathcal{N}(r^{*\top} d, d^\top \Sigma d)$. To prevent the policy π from deviating too much from the reference policy π_0 , we constrain d to lie within a feasible set $\mathcal{D} \subset \mathbb{R}^{XY}$.

Lower Bound on the True Objective Function The following theorem provides a bound on the optimization problem that accounts for the uncertainty in the reward estimates. The proof is presented in Appendix 6.

Theorem 2.2. *Under Assumption 2.1, for any $\beta > 0$, the following holds with probability at least $1 - \exp\left(-\frac{\chi_A}{\beta^2}\right)$:*

$$\sup_{d \in \mathcal{D}} \widehat{R}^\top d - \beta \|d\|_\Sigma \leq \sup_{d \in \mathcal{D}} r^{*\top} d.$$

The above theorem implies that the optimization problem on the left-hand side is a high-probability lower bound for the true optimization problem, which depends on the unknown reward function r^* . Given that r^* is not directly available, but we do have access to noisy estimates \widehat{R} , we propose the following optimization problem as a practical substitute:

$$\sup_{d \in \mathcal{D}} \widehat{R}^\top d - \beta \|d\|_\Sigma. \quad (2)$$

This formulation leads to the following constrained optimization problem:

$$\max_{\pi} \widehat{R}^\top \pi \quad \text{subject to} \quad (\pi - \pi_0)^\top \Sigma (\pi - \pi_0) \leq \varepsilon,$$

for some $\varepsilon > 0$. The weighted constraint on the policy update penalizes deviations more heavily for prompt-response pairs with higher variance in the reward estimates, thereby incorporating the uncertainty into the optimization process.

Remark 2.3. Note that similar variants of the constrained optimization problem have been explored previously in the literature. For example, the unconstrained version of our approach is equivalent to the vanilla policy gradient method (Sutton et al., 1999). The standard RLHF formulation typically employs the PPO algorithm (Schulman et al., 2017), which is defined with a KL-divergence constraint, although the choice of distance metric is not unique. For example, an ℓ_2 approximation of the KL-divergence constraint, resulting in the unweighted constraint: $\|\pi - \pi_0\|_2^2 \leq \varepsilon$. Another widely used technique is the natural policy gradient, as implemented in the Trust Region Policy Optimization (TRPO) algorithm (Schulman, 2015). TRPO adjusts the constraint based on the Fisher information matrix \mathcal{I} , leading to the constraint: $(\pi - \pi_0)^\top \mathcal{I} (\pi - \pi_0) \leq \varepsilon$, where \mathcal{I} adapts the penalization according to the sensitivity of the policy.

In our experiments, we use a variance-adjusted KL-divergence constraint:

$$\mathbb{E}_{x \sim \mathcal{D}, y \sim \pi(\cdot|x)} \left[\sigma^2(x, y) \ln \frac{\pi(y|x)}{\pi_0(y|x)} \right] \leq \varepsilon.$$

This formulation integrates seamlessly with existing PPO subroutines, such as those provided in the **TRL** Library (von Werra et al., 2020)¹.

¹TRL package from **Hugging Face**

3 Theoretical Analysis

We compare the performance of the variance-aware LLM alignment methodology with its variance-unaware counterpart to evaluate how incorporating reward estimate uncertainty affects policy robustness and effectiveness, especially in scenarios with noisy reward estimates. We consider two policies, π_1 and π_2 , derived from different optimization formulations.

Definition 3.1 (Variance-Unaware Policy, π_1). The policy obtained by solving the unweighted ℓ_2 constraint problem:

$$\pi_1 = \arg \max_{\pi} \pi^\top \widehat{R} \quad \text{subject to} \quad \|\pi - \pi_0\|_2^2 \leq \varepsilon.$$

Definition 3.2 (Variance-Aware Policy, π_2). The policy obtained by solving the variance weighted ℓ_2 constraint problem:

$$\pi_2 = \arg \max_{\pi} \pi^\top \widehat{R} \quad \text{subject to} \quad \|\pi - \pi_0\|_{\Sigma}^2 \leq \tilde{\varepsilon}.$$

To compare both methods fairly, we set $\tilde{\varepsilon} = \lambda_{\min}(\Sigma) \cdot \varepsilon$; this has the effect of aligning the largest ellipsoid of the covariance-weighted constraint with the sphere of the traditional ℓ_2 constraint.

Main Result We evaluate the expected true rewards $\pi_i^\top r^*$ for $i = 1, 2$, where r^* is the true (unknown) reward vector for both methods and compare them to $\pi_0^\top r^*$. We aim to show that π_2 is less likely to underperform relative to π_0 than π_1 , indicating that the variance-aware method is less risky when reward estimates are uncertain.

Theorem 3.3. Consider policies π_1 and π_2 as defined in Definitions 3.1 and 3.2 respectively. With $\tilde{\varepsilon}$ set as $\lambda_{\min}(\Sigma)\varepsilon$ to ensure the optimization domain of the variance-aware method is only as large as the variance unaware method, we have the following result:

$$\mathbb{P}(\pi_2^\top r^* \leq \pi_0^\top r^*) \leq \mathbb{P}(\pi_1^\top r^* \leq \pi_0^\top r^*).$$

Remark 3.4. Thus, the variance-aware method (π_2) has a lower probability of underperforming relative to π_0 than the variance-unaware method (π_1). Theorem 3.3 highlights the trade-off between risk and reward. While the variance-unaware policy (π_1) may achieve higher rewards when \widehat{R} is accurate, it is riskier as it ignores estimate uncertainty. The variance-aware policy (π_2) reduces underperformance risk by accounting for reward estimate variance. The proof of the theorem is presented in Appendix 6.

Remark 3.5. Our variance-aware policy is closely related to another reward-to-variability ratio known in finance literature as the Sharpe Ratio (Sharpe, 1966), which balances expected return against risk.

Theorem 3.6. Consider the optimization problem:

$$\begin{aligned} \max_{\pi} \quad & \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi(\cdot|x)} \left[\widehat{R}(x, y) \right] \\ \text{subject to} \quad & \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi(\cdot|x)} \left[\sigma^2(x, y) \ln \frac{\pi(y|x)}{\pi_0(y|x)} \right] \leq \varepsilon, \end{aligned}$$

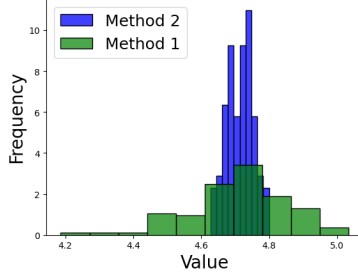


Figure 3: In the high-variability setting, variances of reward estimates range between (3,100). Method 2 (variance-aware) exhibits significantly lower return variance than Method 1 (variance-unaware), confirming its risk-averse nature. The standard deviation for Method 2 is 0.04, while for Method 1 it is 0.13. The mean returns for both methods are comparable: 4.643 for Method 1 and 4.644 for Method 2.

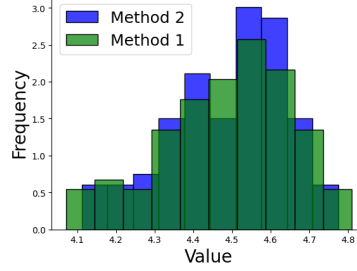


Figure 4: In the low-variability setting, variances of reward estimates range between (70,100). Both methods perform similarly, with Method 2 (variance-aware) having a standard deviation of 0.12 and Method 1 (variance-unaware) having a standard deviation of 0.14. The mean returns for Method 1 and Method 2 are 0.14 and 0.13, respectively.

Figure 5: Distribution of policy returns under different variability settings. In both cases, the true reward vector r^* is fixed, and reward estimates \hat{R} are sampled from a multivariate Gaussian distribution with the specified covariance matrices. The histograms show the frequency of policy returns under both methods, illustrating the risk-averse nature of Method 2 in the high-variability setting and the convergence of both methods in the low-variability setting.

where $\hat{R}(x, y)$ and $\sigma^2(x, y)$ are the reward estimate and its variance. The optimal policy is:

$$\pi^*(y|x) \propto \pi_0(y|x) \exp \left(\frac{\hat{R}(x, y)}{\beta \sigma^2(x, y)} \right),$$

for some $\beta > 0$.

The proof is presented in Appendix 6. Thus, the optimal policy is proportional to $\frac{\hat{R}(x, y)}{\sigma^2(x, y)}$, which is also known as the Sharpe Ratio, which measures the return of an investment after adjusting for its risk.

Variability in the Variance The variance-aware method’s advantages are more significant when reward estimate variances vary across prompt-response pairs. If variances are homogeneous, both methods perform similarly since the covariance-weighted constraint becomes proportional to the traditional ℓ_2 constraint. We conduct simulations to illustrate the benefits of the variance-aware method. We fix a true reward vector r^* (dimension 1000) and sample reward estimates \hat{R} from $\mathcal{N}(r^*, \Sigma)$ under two settings: high and low variance variability. In the high-variability setting (Figure 3), the variance-aware method (π_2) shows significantly lower return variance compared to the

variance-unaware method (π_1), confirming its risk-averse nature. In the low-variability setting (Figure 4), both methods perform similarly, aligning with theoretical predictions. These results confirm our theoretical insights and demonstrate the practical utility of variance-aware policy optimization in aligning LLMs with human preferences.

4 Reward Modeling

In this section, we discuss the process of reward modeling using the **Gemma-2B-it** model (Team et al., 2024), an instruction-tuned version of the foundational model **Gemma-2B**. Our reward modeling methodology uses an ensemble of models, specifically 10 independent reward models, to compute the reward variance across different instances of the same prompt-response pair. This ensemble-based approach allows us to better capture the uncertainty in the reward estimates and to analyze the variability between otherwise identical reward models. The following paragraphs detail the methodology used to learn the ensemble of reward models, the dataset used for training and evaluation, and the observations drawn from the ensemble’s performance across multiple benchmarks.

Dataset To train our reward models, we utilize an existing open-source preference dataset (Dong et al., 2024), which is available publicly via HuggingFace². This curated dataset contains approximately 50,000 labeled preference pairs. It is constructed by combining several well-known, open-source datasets. The included datasets are **HH-RLHF** (Bai et al., 2022a), **SHF** (Ethayarajh et al., 2022), **HelpSteer** (Wang et al., 2023), **PKU-SafeRLHF** (Ji et al., 2024), **UltraFeedback** (Cui et al., 2023), **UltraInteract** (Yuan et al., 2024), **Distilabel-Capybara** (Daniele, 2023), and **Distilabel-Orca3** (Lian et al., 2023). The combined dataset has undergone preprocessing to filter out sub-quality data, specifically removing 10% of the original dataset to ensure the quality of the training samples. The final dataset contains human preferences where, for each prompt, two responses are given: one preferred and the other rejected. The preference labels serve as the ground truth for training our ensemble of reward models. This dataset provides a comprehensive and diverse set of prompt-response pairs, making it suitable for training a robust reward model ensemble that can generalize across various domains and tasks. We refer readers to the original work of Dong et al. (2024) for further details on the dataset construction and preprocessing steps.

Methodology We use the **Gemma-2B-it** (Gemma, 2024) model as the foundation for our reward models. The instruction-tuned nature of this model makes it a strong candidate for reward modeling tasks, as it has been fine-tuned to follow human instructions closely. The size of **Gemma-2B-it** is approximately 9.34 GB on disk, including a scalar reward head. Given that we use an ensemble of 10 independent reward models, the total storage required for all models is approximately 90 GB. To accelerate the training process and optimize memory usage, we employ the following methodology:

²huggingface.co/weqweasdas/preference_dataset_mix2

Model	Average Score	Chat	Chat-Hard	Safety	Reasoning	Prior Sets
GRM-Gemma-2B-sftreg (Yang et al., 2024)	74.7	95.5	48.7	80.0	76.8	69.8
Gemma-2B-rewardmodel-baseline	73.1	94.1	46.9	79.7	73.8	69.0
Our Model	69.4	95.6	44.5	55.9	81.8	69.0
Qwen1.5-72B-Chat (Bai et al., 2023)	68.2	62.3	66.0	72.0	85.5	42.3
MiniCPM-2B-dpo-fp32 (Hu et al., 2024)	66.2	89.1	49.3	52.5	82.3	49.6
RM-Gemma-2B (Dong et al., 2023)	64.2	94.4	40.8	44.0	76.4	66.5

Table 1: Comparison of our ensemble of reward models to other SOTA 2B models on the **RewardBench** platform. The **Prior Sets** are given 50% weightage in the final score. Our model shows competitive performance compared to others, highlighting its efficacy in reward modeling tasks.

- **Initial Training:** We begin by training a single instance of the full **Gemma-2B-it** model with a scalar reward head on the preference dataset. The reward head is a simple linear layer with dimensions 2048×1 . We use *early-stopping* during training to prevent overfitting and ensure generalization. Specifically, we stop training when the loss reaches 0.3, as this strikes a balance between model complexity and the risk of overfitting.
- **Parallel Reward Heads:** Once the initial model is partially trained, we attach 9 additional reward heads in parallel with the original reward head (Zhang et al., 2024). Each reward head is a linear layer with the same dimensions as the first (2048×1). The model now outputs a 10-dimensional vector, where each element corresponds to the reward output of one of the 10 models in the ensemble. This configuration allows us to efficiently compute the rewards for all models in a single forward pass.
- **Freezing the Foundation Model:** To reduce computational complexity and ensure faster training, we freeze the weights of the foundation model (i.e., the pre-trained layers of **Gemma-2B-it**) and train only the reward heads. This allows us to simulate training 10 independent reward models in parallel while sharing the foundation model across all reward heads. We employ an additive loss function during training: $\text{loss} = \sum_{i=1}^{10} l(\theta_i)$, where each θ_i represents the parameters of the i -th reward head. This approach ensures that all reward heads are trained independently but computationally efficiently. In this sense, our methodology differs from the one used in Zhang et al. (2024).

By freezing the foundational layers and focusing the training on the reward heads, we can significantly reduce the computational and storage costs associated with training an ensemble of models. The final ensemble model occupies approximately 9.34 GB on disk, and the total number of trainable parameters across all reward heads is 20,480.

Evaluation To assess the performance of our ensemble reward models, we utilize the **RewardBenchmark** platform (Lambert et al., 2024)³, a widely-used platform that offers curated datasets and evaluation metrics specifically designed for benchmarking reward models. This platform provides an in-depth evaluation across multiple datasets, each designed to test different aspects of reward modeling, such as conversational ability, safety, and reasoning. The evaluation is conducted on four primary datasets: **Chat**

³<https://huggingface.co/allenai/reward-bench>

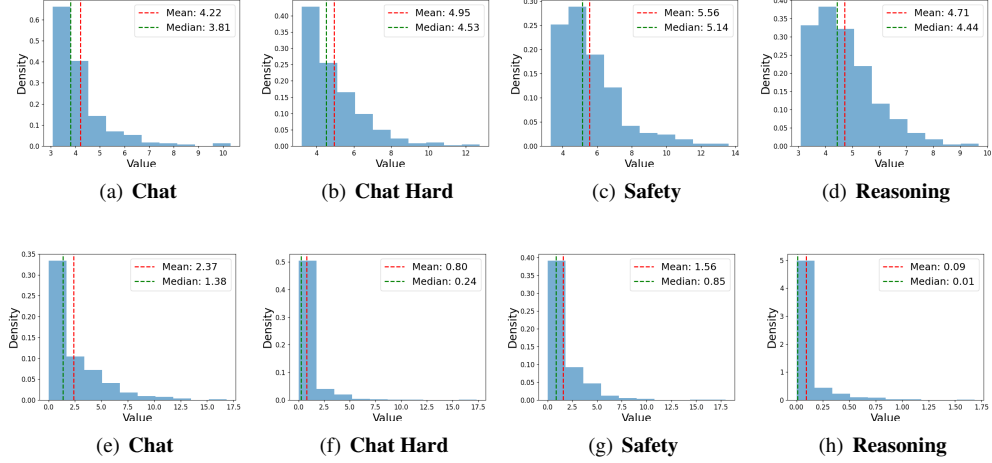


Figure 6: (*Top Row*) The distribution of sample variances of the reward on the accepted responses. The 10 reward models calculate the sample variance. We note from the median of the sample variances that half of the dataset tends to have variances of the rewards greater than 3.81, with a maximum close to 10. This corroborates our hypothesis that different reward models will exhibit variability in their reward assignments for the same prompt-response pair. (*Bottom Row*) The distribution of sample variance of the rewards difference between accepted and rejected responses. The figure shows that the reward models are not merely translations of one another, and the variance arises due to the statistical nature of learning these reward models and the stochasticity of the optimization process.

(Li et al., 2023; Zheng et al., 2023), **Chat-Hard** (Zheng et al., 2023), **Safety** (Röttger et al., 2023; Dong et al., 2023), and **Reasoning** (Muennighoff et al., 2023; Lightman et al., 2023). Additionally, there is a fifth dataset called **Prior**, which consists of subsets of various other datasets including **Anthropic Helpful** (Bai et al., 2022a), **BIG-Bench** (Askell et al., 2021), **Stanford Human Preferences (SHP)** (Ethayarajh et al., 2022) and **Learning to Summarize** (Stiennon et al., 2020) and is given a 50% weightage in the overall score. The platform evaluates models based on a comprehensive list of metrics, providing a holistic view of the model’s ability to predict human preferences. We refer readers to the original work for a more detailed explanation of the dataset composition. We compare the average performance of our ensemble model to other state-of-the-art (SOTA) models with similar model sizes (2B parameters). Table 1 summarizes the results of this comparison. Our ensemble reward model demonstrates performance comparable to other SOTA 2B models, confirming its efficacy as a reliable reward estimation framework.

Observations To corroborate our hypothesis that identically trained reward models disagree on the same prompt-response pair, we run our experiment on the 4 datasets provided in the **RewardBenchmark** platform, namely **Chat**, **Chat-Hard**, **Safety** and **Reasoning** datasets. For example, the **Chat** dataset contains 358 prompt-response pairs in the form (x, y^1, y^2) , where y^1 is the accepted response, and y^2 is the rejected response. The **Chat** dataset is a mixture of multiple sources, including **AlpacaEval**

Easy, **AlpacaEval**, **AlpacaEval Hard** (Li et al., 2023), **MT Bench Easy**, and **MT Bench Medium** (Zheng et al., 2023). The composition of the other datasets can be found in the original work of Lambert et al. (2024). We analyze the variance of the rewards assigned to the accepted responses across the 10 models in the ensemble. For each prompt x , we compute the reward for the accepted response $r_i(x, y^1)$ using the i -th reward model. We continue to compute the sample variance of the rewards for each accepted response across the 10 models and plot the distribution of the sample variance of the entire dataset. The top row of Figure 6 shows the histogram of the computed sample variances in each dataset. We observe that the variances in the rewards range between 3 and 14, with a mean variance greater than 4 and a median variance greater than 3 for each dataset. This indicates that there is non-negligible variability in the rewards assigned by the different models in the ensemble, even though the models are trained on the same dataset. This lack of uniformity can be attributed to factors such as the finite size of the training data and the inherent stochasticity of the optimization process used during training. These findings align with our hypothesis that different reward models can exhibit notable disagreement in their reward assignments for the same prompt-response pair, even when trained on identical data. To further explore this variability, we analyze the variance distribution of the differences between the rewards assigned to the accepted and rejected responses. The bottom row of Figure 6 presents this distribution, illustrating that the reward models are not simply translations of one another. Translationally invariant models would exhibit no differences in rewards, leading to a Dirac distribution centered at zero. However, the distribution as observed shows that this is not the case, supporting the notion that the observed variance arises from the statistical and stochastic nature of the learning process.

5 Proximal Policy Optimization (PPO)

This section describes our methodology for fine-tuning the **GPT-2** (Radford et al., 2019) language model using a variance-aware approach. Our approach builds on the standard Proximal Policy Optimization (PPO) framework (Schulman et al., 2017), modified to incorporate uncertainty in the reward estimates. The goal is to demonstrate how accounting for variance in reward models can lead to more robust and safe policies. We note that the reason for choosing **GPT-2** was based on the ease of performing PPO, as it is known in the literature that training large language models with PPO presents difficulties involving instability and sensitivity to hyperparameters (Choshen et al., 2019), code-level optimizations (Engstrom et al., 2020) and resource intensiveness.

Dataset For prompt sampling, we use the **IMDB** dataset (Maas et al., 2011), which is publicly available via Hugging Face⁴. The train split of this dataset consists of 25,000 rows. We sample prompts x from each row with random lengths between 2 to 8 tokens. These sampled prompts serve as input to the language model during the training process, where responses are generated and evaluated by our reward models.

⁴stanfordnlp/imdb

Methodology We use **GPT-2** as the base language model for fine-tuning. The responses generated by **GPT-2** have a maximum length of 10 tokens. For each prompt-response pair (x, y) , we compute rewards and variances from each of the 10 reward models in our ensemble. The reward for a given pair is adjusted by penalizing the score based on the variance-weighted KL divergence between the current policy π and the reference policy; that is, the adjusted reward is given by: $R_i(x, y) = r_i(x, y) - \beta \sigma(x, y) \ln \frac{\pi(y|x)}{\pi_0(y|x)}$, where $r_i(x, y)$ is the reward from the i -th model. Note that this estimate differs from the lower confidence estimate $r_i - \beta \sigma$ used in previous works (Zhang et al., 2024). Using this variance-weighted reward, we perform PPO to update the policy. For each reward model, we run 4 independent trials of PPO, resulting in 4 policies per reward model. We train 40 independent policies, which we label as the *variance-aware* policies. These policies are compared with another set of policies trained using the conventional PPO method as given in **TRL** library (von Werra et al., 2020). To ensure a fair comparison between the two methods, we fine-tune the value of β experimentally to equalize the KL divergence between the final policy and the reference policy across both sets of policies.

Evaluation To assess the quality of the trained policies, we evaluate them using a large reward model that serves as a judge. Specifically, we use the **FsfairX-LLaMA3-RM-v0.1** reward model (Dong et al., 2023; Xiong et al., 2024)⁵, which is based on **LLama-3-8B** and currently ranks 17 on the **RewardBenchmark** platform. This reward model acts as an evaluator by scoring the prompt-response pairs generated by the trained policies. Each of the 40 policies from the *variance-aware* set is used to generate responses for the test split of the **IMDB** dataset. The responses are then evaluated by the judge reward model, which assigns an average score for the entire test dataset. This process results in a distribution of average rewards for the *variance-aware* policies. We repeat the same evaluation for the *vanilla-PPO* policies, generating another reward distribution based on their performance. As a baseline, we also evaluate the performance of the reference policy, **GPT-2**, using the same reward model. The reward distributions for all three sets of policies are compared and plotted in Figure 7.

Observations In Figure 7, indigo marks the true reward distribution of the base or reference policy of **GPT-2** as measured by the judge reward model. The red marks the true reward distribution of the variance-aware policy, while the cyan marks the true reward distribution of the vanilla PPO policy. As can be seen from the figure, the mean reward of both methods performs better than the reference policy, which has a mean reward of 0.19. The *Variance-Aware Policy* shows an improvement over the reference policy, with a mean reward of 0.22 and a variance of 0.012. These policies are trained to be more conservative, which leads to a more robust, albeit less aggressive, improvement in the reward scores. The vanilla PPO policy demonstrates the highest average reward, with a mean of 0.34 but also a significantly higher variance of 0.06. This suggests that while ignoring variance in the reward model can result in larger potential gains, it comes with increased variability and risk, making these policies more

⁵<https://huggingface.co/sfairXC/FsfairX-LLaMA3-RM-v0.1>

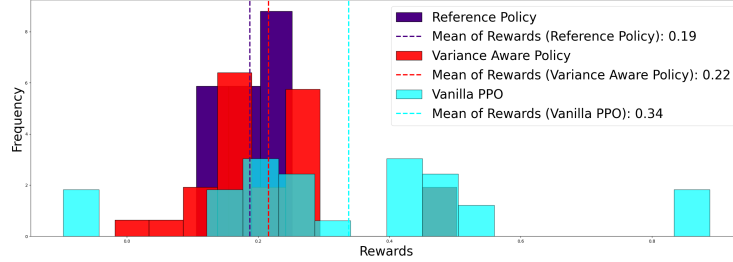


Figure 7: The reward distribution for the two methods compared with the reference policy’s quality. The distribution marked in indigo represents the reward distribution for the reference policy, based on 40 samples of the average reward determined by the judge reward model on responses generated by **GPT-2**. The reward distribution from the reference policy has a mean of 0.19 and a variance of 0.002. The reward distribution for the variance-aware method (in red) has a mean of 0.22 and a variance of 0.012. The reward distribution for the vanilla PPO method (in cyan) has a mean of 0.34 and a variance of 0.06.

sensitive to noise in the reward estimates. The results suggest that the variance-aware approach offers a more stable, risk-averse policy.

References

- Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.
- AI Anthropic. Introducing claude, 2023.
- Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, et al. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*, 2021.
- Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.
- Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022a.
- Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022b.
- Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

- Leshem Choshen, Lior Fox, Zohar Aizenbud, and Omri Abend. On the weaknesses of reinforcement learning for neural machine translation. *arXiv preprint arXiv:1907.01752*, 2019.
- Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.
- Thomas Coste, Usman Anwar, Robert Kirk, and David Krueger. Reward model ensembles help mitigate overoptimization. *arXiv preprint arXiv:2310.02743*, 2023.
- Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. Ultrafeedback: Boosting language models with high-quality feedback. *arXiv preprint arXiv:2310.01377*, 2023.
- Luigi Daniele. Suphavadeeprasit. *Amplify-Instruct: Synthetically Generated Diverse Multi-turn Conversations for Efficient LLM Training*. *arXiv preprint arXiv:(coming soon)*, 2023.
- Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, KaShun SHUM, and Tong Zhang. RAFT: Reward ranked finetuning for generative foundation model alignment. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL <https://openreview.net/forum?id=m7p507zblY>.
- Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. Rlh workflow: From reward modeling to online rlhf. *arXiv preprint arXiv:2405.07863*, 2024.
- Jacob Eisenstein, Chirag Nagpal, Alekh Agarwal, Ahmad Beirami, Alex D’Amour, DJ Dvijotham, Adam Fisch, Katherine Heller, Stephen Pfohl, Deepak Ramachandran, et al. Helping or herding? reward model ensembles mitigate but do not eliminate reward hacking. *arXiv preprint arXiv:2312.09244*, 2023.
- Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep policy gradients: A case study on ppo and trpo. *arXiv preprint arXiv:2005.12729*, 2020.
- Kawin Ethayarajh, Yejin Choi, and Swabha Swayamdipta. Understanding dataset difficulty with v-usable information. In *International Conference on Machine Learning*, pages 5988–6008. PMLR, 2022.
- Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, 2016.
- Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. PMLR, 2023.

- Gemma. <https://huggingface.co/google/gemma-2b>, 2024.
- Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.
- Jiaming Ji, Mickel Liu, Josef Dai, Xuehai Pan, Chi Zhang, Ce Bian, Boyuan Chen, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *Advances in Neural Information Processing Systems*, 36, 2024.
- Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Ensemble-based uncertainty estimation for deep learning. *arXiv preprint arXiv:1612.01474*, 2016.
- Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. Rewardbench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*, 2024.
- Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. AlpacaEval: An automatic evaluator of instruction-following models, 2023.
- W Lian, B Goodson, E Pentland, et al. Openorca: An open dataset of gpt augmented flan reasoning traces, 2023.
- Xinran Liang, Katherine Shu, Kimin Lee, and Pieter Abbeel. Reward uncertainty for exploration in preference-based reinforcement learning. *arXiv preprint arXiv:2205.12401*, 2022.
- 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.
- Xingzhou Lou, Dong Yan, Wei Shen, Yuzi Yan, Jian Xie, and Junge Zhang. Uncertainty-aware reward model: Teaching reward models to know what is unknown. *arXiv preprint arXiv:2410.00847*, 2024.
- Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/P11-1015>.
- AI Meta. Introducing meta llama 3: The most capable openly available llm to date. *Meta AI*, 2024.

- Niklas Muennighoff, Qian Liu, Armel Zebaze, Qinkai Zheng, Binyuan Hui, Terry Yue Zhuo, Swayam Singh, Xiangru Tang, Leandro Von Werra, and Shayne Longpre. Octopack: Instruction tuning code large language models. *arXiv preprint arXiv:2308.07124*, 2023.
- R OpenAI. Gpt-4 technical report. arxiv 2303.08774. *View in Article*, 2(5), 2023.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.
- Alexandre Ramé, Nino Vieillard, Léonard Hussenot, Robert Dadashi, Geoffrey Cideron, Olivier Bachem, and Johan Ferret. Warm: On the benefits of weight averaged reward models. *arXiv preprint arXiv:2401.12187*, 2024.
- Paul Röttger, Hannah Rose Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. Xstest: A test suite for identifying exaggerated safety behaviours in large language models. *arXiv preprint arXiv:2308.01263*, 2023.
- John Schulman. Trust region policy optimization. *arXiv preprint arXiv:1502.05477*, 2015.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- William F Sharpe. Mutual fund performance. *The Journal of business*, 39(1):119–138, 1966.
- Lingfeng Shen, Sihao Chen, Linfeng Song, Lifeng Jin, Baolin Peng, Haitao Mi, Daniel Khashabi, and Dong Yu. The trickle-down impact of reward (in-) consistency on rlhf. *arXiv preprint arXiv:2309.16155*, 2023.
- Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.
- Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

- Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.
- Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Galouédec. Trl: Transformer reinforcement learning. <https://github.com/huggingface/trl>, 2020.
- Zhilin Wang, Yi Dong, Jiaqi Zeng, Virginia Adams, Makesh Narsimhan Sreedhar, Daniel Egert, Olivier Delalleau, Jane Polak Scowcroft, Neel Kant, Aidan Swope, et al. Helpsteer: Multi-attribute helpfulness dataset for steerlm. *arXiv preprint arXiv:2311.09528*, 2023.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL <https://www.aclweb.org/anthology/2020.emnlp-demos.6>.
- Wei Xiong, Hanze Dong, Chenlu Ye, Ziqi Wang, Han Zhong, Heng Ji, Nan Jiang, and Tong Zhang. Iterative preference learning from human feedback: Bridging theory and practice for rlhf under kl-constraint. *ICML*, 2024.
- Rui Yang, Ruomeng Ding, Yong Lin, Huan Zhang, and Tong Zhang. Regularizing hidden states enables learning generalizable reward model for llms. *arXiv preprint arXiv:2406.10216*, 2024.
- Lifan Yuan, Ganqu Cui, Hanbin Wang, Ning Ding, Xingyao Wang, Jia Deng, Boji Shan, Huimin Chen, Ruobing Xie, Yankai Lin, et al. Advancing llm reasoning generalists with preference trees. *arXiv preprint arXiv:2404.02078*, 2024.
- Yuanzhao Zhai, Han Zhang, Yu Lei, Yue Yu, Kele Xu, Dawei Feng, Bo Ding, and Huaimin Wang. Uncertainty-penalized reinforcement learning from human feedback with diverse reward lora ensembles. *arXiv preprint arXiv:2401.00243*, 2023.

Shun Zhang, Zhenfang Chen, Sunli Chen, Yikang Shen, Zhiqing Sun, and Chuang Gan. Improving reinforcement learning from human feedback with efficient reward model ensemble. *arXiv preprint arXiv:2401.16635*, 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.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

6 Proofs

Theorem 2.2. *Under Assumption 2.1, for any $\beta > 0$, the following holds with probability at least $1 - \exp\left(-\frac{\mathbf{X}\mathbf{A}}{\beta^2}\right)$:*

$$\sup_{d \in \mathbf{D}} \widehat{R}^\top d - \beta \|d\|_\Sigma \leq \sup_{d \in \mathbf{D}} r^{*\top} d.$$

Proof. The result follows from a standard self-normalizing bound for Gaussian random variables. Specifically, for any $\delta > 0$, the following inequality holds with high probability:

$$\left\| \widehat{R} - r^* \right\|_{\Sigma^{-1}} \leq \sqrt{\mathbf{X}\mathbf{A} \ln(1/\delta)},$$

with probability at least $1 - \delta$, since $\left\| \widehat{R} - r^* \right\|_{\Sigma^{-1}}$ is the self-normalized euclidean norm of a standard Gaussian random variable in $\mathbf{X}\mathbf{A}$ dimensions. By applying the Cauchy-Schwarz inequality, we have, for any $d \in \mathbf{D}$:

$$\left| \langle d, \widehat{R} - r^* \rangle \right| \leq \|d\|_\Sigma \left\| \widehat{R} - r^* \right\|_{\Sigma^{-1}}.$$

Substituting the bound on $\left\| \widehat{R} - r^* \right\|_{\Sigma^{-1}}$, we obtain:

$$\left| \langle d, \widehat{R} - r^* \rangle \right| \leq \|d\|_\Sigma \sqrt{\mathbf{X}\mathbf{A} \ln(1/\delta)}.$$

This completes the proof. \square

Theorem 3.3. *Consider policies π_1 and π_2 as defined in Definitions 3.1 and 3.2 respectively. With $\tilde{\varepsilon}$ set as $\lambda_{\min}(\Sigma)\varepsilon$ to ensure the optimization domain of the variance-aware method is only as large as the variance unaware method, we have the following result:*

$$\mathbb{P}\left(\pi_2^\top r^* \leq \pi_0^\top r^*\right) \leq \mathbb{P}\left(\pi_1^\top r^* \leq \pi_0^\top r^*\right).$$

Proof. Both optimization problems ((3.1) and (3.2)) involve maximizing a linear function over a convex domain. Thus, the maximum occurs at the boundary of the feasible region, allowing us to replace the inequality constraints in (3.1) and (3.2) with equality constraints. We can solve these optimization problems using the method of Lagrange multipliers. For the variance-aware optimization problem (3.2), the Lagrangian formulation is:

$$\pi_2 = \operatorname{argmax}_{\pi} \left[\widehat{R}^\top \pi - \beta (\pi - \pi_0)^\top \Sigma (\pi - \pi_0) \right],$$

where β is the Lagrange multiplier associated with the covariance-weighted ℓ_2 constraint. The solution to this optimization problem is given by:

$$\pi_2 = \pi_0 + \frac{1}{2\beta} \Sigma^{-1} \widehat{R}. \quad (3)$$

To satisfy the constraint $\|\pi_2 - \pi_0\|_\Sigma^2 = \tilde{\varepsilon}$, we determine β as:

$$\beta = \frac{1}{2} \sqrt{\frac{\widehat{R}^\top \Sigma^{-1} \widehat{R}}{\tilde{\varepsilon}}}.$$

Substituting this back into the solution for π_2 yields:

$$\pi_2 = \pi_0 + \sqrt{\frac{\tilde{\varepsilon}}{\widehat{R}^\top \Sigma^{-1} \widehat{R}}} \Sigma^{-1} \widehat{R}.$$

Similarly, for the variance-unaware policy π_1 , solving the optimization problem (3.1) yields:

$$\pi_1 = \pi_0 + \sqrt{\frac{\varepsilon}{\widehat{R}^\top \widehat{R}}} \widehat{R}.$$

Next, we compute the expected true rewards under both policies. The true reward under π_1 is:

$$\pi_1^\top r^* = \pi_0^\top r^* + \sqrt{\frac{\varepsilon}{\widehat{R}^\top \widehat{R}}} \widehat{R}^\top r^*,$$

and under π_2 , the true reward is:

$$\pi_2^\top r^* = \pi_0^\top r^* + \sqrt{\frac{\tilde{\varepsilon}}{\widehat{R}^\top \Sigma^{-1} \widehat{R}}} \widehat{R}^\top \Sigma^{-1} r^*.$$

Both policies underperform relative to π_0 if their corresponding rewards are less than or equal to $\pi_0^\top r^*$. For π_1 , this occurs if $\widehat{R}^\top r^* \leq 0$, and for π_2 , this occurs if $\widehat{R}^\top \Sigma^{-1} r^* \leq 0$. Since \widehat{R} is normally distributed with mean r^* and covariance Σ , we have:

$$\begin{aligned} \widehat{R}^\top r^* &\sim \mathcal{N}(\|r^*\|^2, r^{*\top} \Sigma r^*), \\ \widehat{R}^\top \Sigma^{-1} r^* &\sim \mathcal{N}(r^{*\top} \Sigma^{-1} r^*, r^{*\top} \Sigma^{-1} r^*). \end{aligned}$$

Thus, the probabilities of underperformance are given by:

$$\begin{aligned}\mathbb{P}\left(\widehat{R}^\top r^* \leq 0\right) &= \Phi\left(-\frac{\|r^*\|^2}{\sqrt{r^{*\top}\Sigma r^*}}\right), \\ \mathbb{P}\left(\widehat{R}^\top \Sigma^{-1} r^* \leq 0\right) &= \Phi\left(-\sqrt{r^{*\top}\Sigma^{-1} r^*}\right),\end{aligned}$$

where Φ is the standard normal cumulative distribution function. Using the Cauchy-Schwarz inequality:

$$\begin{aligned}\|r^*\|^2 &= r^{*\top}\Sigma^{-1/2}\Sigma^{1/2}r^* \\ &\leq \left\|\Sigma^{-1/2}r^*\right\| \left\|\Sigma^{1/2}r^*\right\| \\ &= \sqrt{r^{*\top}\Sigma^{-1}r^*} \sqrt{r^{*\top}\Sigma r^*}.\end{aligned}$$

Thus, we conclude:

$$-\frac{\|r^*\|^2}{\sqrt{r^{*\top}\Sigma r^*}} \geq -\sqrt{r^{*\top}\Sigma^{-1}r^*}.$$

Since the cumulative distribution function Φ is increasing, it follows that:

$$\mathbb{P}\left(\pi_2^\top r^* \leq \pi_0^\top r^*\right) \leq \mathbb{P}\left(\pi_1^\top r^* \leq \pi_0^\top r^*\right).$$

□

Theorem 3.6. *Consider the optimization problem:*

$$\begin{aligned}\max_{\pi} \quad & \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi(\cdot|x)} \left[\widehat{R}(x, y) \right] \\ \text{subject to} \quad & \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi(\cdot|x)} \left[\sigma^2(x, y) \ln \frac{\pi(y|x)}{\pi_0(y|x)} \right] \leq \varepsilon,\end{aligned}$$

where $\widehat{R}(x, y)$ and $\sigma^2(x, y)$ are the reward estimate and its variance. The optimal policy is:

$$\pi^*(y|x) \propto \pi_0(y|x) \exp\left(\frac{\widehat{R}(x, y)}{\beta \sigma^2(x, y)}\right),$$

for some $\beta > 0$.

Proof. The constrained optimization problem can be transformed into an unconstrained optimization problem by introducing a Lagrange multiplier $\beta > 0$:

$$\operatorname{argmax}_{\pi} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi(\cdot|x)} \left[\frac{\widehat{R}(x, y)}{\beta \sigma^2(x, y)} - \ln \frac{\pi(y|x)}{\pi_0(y|x)} \right].$$

The proof follows standard techniques and can be found in Rafailov et al. (2024) (Appendix A.1). □

7 Experimental Details for Reward Modeling

The hyperparameter details used in the single reward-head modeling are given in Table 2. Other parameters are kept as in Wolf et al. (2020). Table 3 summarizes the hardware specifications and resource consumption during the single reward-head training process, including GPU memory, disk space, and total training time. The model is trained using four NVIDIA A40 GPUs, each with 48 GB of memory. The total disk space for storing the dataset, model checkpoints, and logs is approximately 30 GB. Training time is 51 hours.

Hyperparameter	Value
Effective Batch Size	32
Learning Rate	1e-5
Optimizer	Paged AdamW 32bit
Weight Decay	0.001
LR Scheduler	cosine
Epochs	1
Global Train Steps	4125

Table 2: Hyperparameters used in training the Single Reward Model.

Resource	Details
GPU Model	NVIDIA A40 (40 GB)
Number of GPUs	4
Total GPU Memory	12.68 GB
Total Disk Space Required	30 GB
Total Training Time	51 hours

Table 3: Hardware requirements for training the single reward model.

The hyperparameter details used in ensemble reward modeling are given in Table 4. Other parameters are kept as in Wolf et al. (2020). Table 5 summarizes the hardware specifications and resource consumption during the ensemble training process, including GPU memory, disk space, and total training time. The model is trained using four NVIDIA A40 GPUs, each with 48 GB of memory. The total disk space for storing the dataset, model checkpoints, and logs is approximately 40 GB. Training time is 7 hours.

Hyperparameter	Value
Effective Batch Size	32
Learning Rate	1e-5
Optimizer	Paged AdamW 32bit
Weight Decay	0.001
LR Scheduler	cosine
Epochs	0.5
Global Train Steps	2060

Table 4: Hyperparameters used in training the Ensemble Reward Model.

Resource	Details
GPU Model	NVIDIA A40
Number of GPUs	4
Total GPU Memory	6.12 GB
Total Disk Space Required	38 GB
Total Training Time	7 hours

Table 5: Hardware requirements for training the ensemble reward model.

Figures 8(a) and 8(b) depict the training loss curves for both the single and ensemble reward models. In particular, we early-stop the fine-tuning of the single reward-head model when the loss dips below the 0.4 mark. We then attach 10 reward heads parallel to the final layer, freeze the base model, and retrain only the reward heads until the average training loss for each reward head is close to 0.2.

In Figure 9, we present the performance of the ten models evaluated across four datasets on the **RewardBenchmark** platform: **Chat**, **Chat-Hard**, **Reasoning**, and **Safety**. In particular, we compare these models against a fully fine-tuned single reward head model instead of the ensemble models trained with a frozen base. Our results

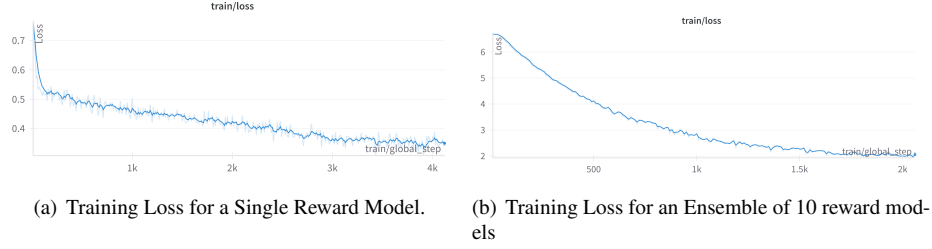


Figure 8: Training Loss for Reward Modelling

indicate that the models within the ensemble perform on par with each other and are comparable to the fully fine-tuned single reward head model.

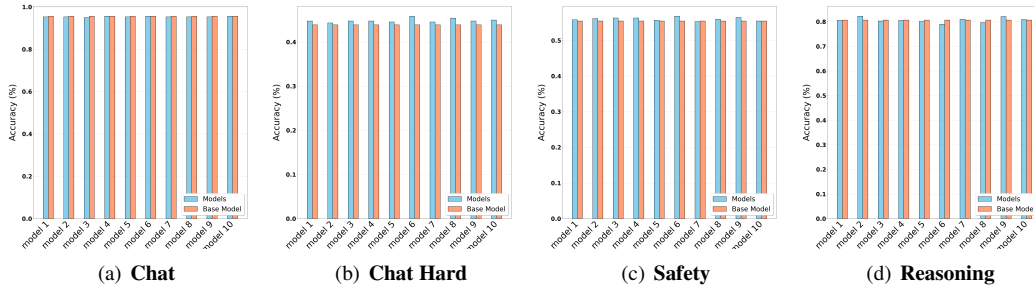


Figure 9: The comparison of each model in the ensemble with the single reward-head model on all evaluation datasets of the **RewardBenchmark** platform. In particular, the 10 blue bars indicate the model accuracy for each of the 10 models. The accuracy of the base model is given in orange. We see that for each of the 10 models in the ensemble, the performance is comparable with the base model.

8 Experimental Details for PPO Training

The hyperparameter and details used in both the vanilla and the variance-aware PPO training are given in Tables 6 and 7. Most of the hyperparameters are taken as in von Werra et al. (2020). The major difference between the two methods is a judicious choice of the β parameter, which controls the constraint domain of the optimization problem. To be consistent, we choose the β parameter such that the KL divergence from the reference policy is roughly the same for both methods. This ensures that the search domains for both methods are roughly the same. The β parameter is defined as the *Initial KL Coeff* variable in the hyperparameter tables.

Table 8 summarizes the hardware specifications and resource consumption for training a single **GPT-2** model using PPO, including GPU memory, disk space, and total training time. The model is trained using four NVIDIA A40 GPUs, each with 48 GB

Hyperparameter	Value
Effective Batch Size	128
Learning Rate	1.414e-5
Epochs	1
Steps	192
Initial KL Coeff	0.2
Adaptive KL Control	False

Table 6: Hyperparameters used in training with vanilla PPO method.

Hyperparameter	Value
Effective Batch Size	128
Learning Rate	1.5e-5
Epochs	1
Steps	192
Initial KL Coeff	0.05
Adaptive KL Control	False

Table 7: Hyperparameters used in training with Variance Aware PPO method.

of memory. The total disk space for storing the dataset, model checkpoints, and logs is approximately 6.55 GB. Training time is roughly 4 hours.

Resource	Details
GPU Model	NVIDIA A40
Number of GPUs	4
Total GPU Memory	18.4 GB
Total Disk Space Required	6.55 GB
Total Training Time	3.86 hours

Table 8: Hardware requirements for training a single PPO model.

Figure 10 shows the evolution of the KL divergence between the trained and reference policies for both methods. The average and standard deviation of the KL divergence for the 40 policies for both sets of methods are plotted. As can be seen with high probability, the KL divergence for both methods lies within the 1.2 and 1.4 range. Each of the 40 independent policies was run with an initial random seed of 0.

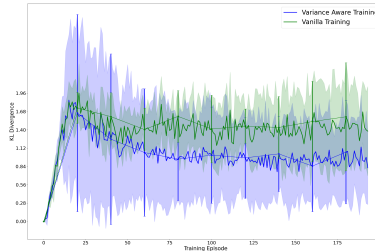


Figure 10: The trajectories of the KL divergence as a function of training steps are plotted for both methods. Specifically, we plot the mean KL and the standard deviation of the KL for the 40 independently trained policies for both methods. Green denotes the KL trajectory for the vanilla PPO method, whereas blue indicates the variance-aware method. As can be seen, by the end of the training, with high probability, the KL divergence of the final policy from the reference policy is roughly the same for both methods. In particular, both methods produce policies whose KL divergences from the reference policy lie between 1.2 and 1.4.

Figure 11 shows the evolution of the rewards collected by the policies for both methods. The average and standard deviation of the rewards for the 40 policies for

both sets of methods are plotted.

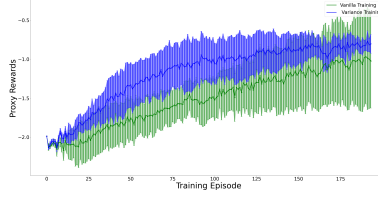


Figure 11: The trajectories of the proxy reward as a function of training steps are plotted for both methods. Specifically, we plot the mean proxy reward and the standard deviation of the proxy rewards for the 40 independently trained policies for both methods. Green denotes the trajectory for the vanilla PPO method, whereas blue indicates the variance-aware method.

In Figure 12, we repeat the experiment of Section 5, but this time with 100 sample policies trained using the vanilla and the variance aware method and evaluated using the judge reward model.

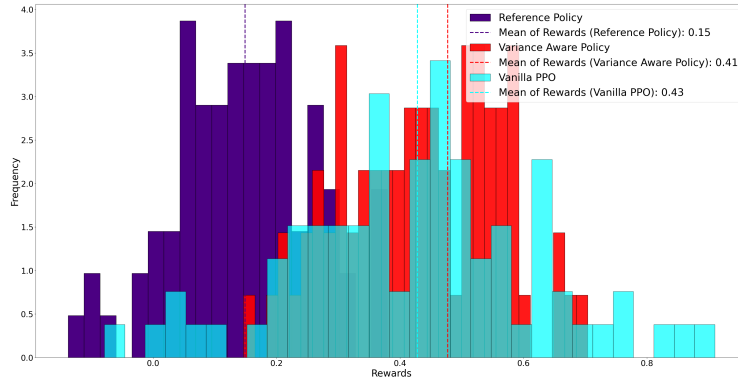


Figure 12: The reward distribution for the two methods compared with the reference policy’s quality. The distribution marked in indigo represents the reward distribution for the reference policy, based on 100 samples of the average reward determined by the judge reward model on responses generated by **GPT-2**. The reward distribution from the reference policy has a mean of 0.15 and a variance of 0.012. The reward distribution for the variance-aware method (in red) has a mean of 0.41 and a variance of 0.016. The reward distribution for the vanilla PPO method (in cyan) has a mean of 0.43 and a variance of 0.038.