# POLICY-AWARE REWARD MODELING WITH UNCERTAINTY-GRADIENT BASED DATA AUGMENTATION

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

## ABSTRACT

Reinforcement Learning from Human Feedback (RLHF) has emerged as a standard and effective approach for training large language models (LLMs) with human preferences. In this framework, a learned reward model approximates human preferences and guides policy optimization, making it crucial to develop an accurate reward model. However, without the "true" reward function, challenges arise when the reward model is an imperfect proxy for human preference. Since the policy optimization continuously shifts the human preference training dataset's distribution. The fixed reward model suffers from this problem of off-distribution, especially the on policy methods. While collecting new preference data can mitigate this issue, it is costly and challenging to optimize. Thus, reusing the policy interaction samples becomes a possible way to further refine the reward model. To tackle these challenges, we introduce a novel method **U**ncertainty-**G**radient based **D**ata **A**ugmentation (**UGDA** for short) to enhance reward modeling by leveraging policy samples to maintain on-distribution performance. Specifically, UGDA selects interaction samples based on the uncertainty of the reward ensembles and the gradient based influence of policy optimization. After the reward relabeling of selected samples, we use supervised learning to refine the reward ensembles, then get the retrained policy. Extensive experiments demonstrate that by leveraging UGDA to select a few samples without the costly human preference data collection, we can improve the ability of the policy and surpass the state-of-the-art methods.

## 1 INTRODUCTION

Large Language Models (LLMs) have demonstrated promising results in comprehending human queries and providing valuable responses (Achiam et al., 2023). Reinforcement Learning from Human Feedback (RLHF) has emerged as a potent technique for fine-tuning LLMs and aligning language model outputs with human preferences (Ziegler et al., 2019; Ouyang et al., 2022). Despite its empirical success, RLHF suffers from many challenges (Casper et al., 2023). The reward model is utilized to approximate human preferences during the training process of RLHF. However, the learned reward model serves as a proxy for the "true" reward function. In line with Goodhart's law, excessive optimization of reward scores may hinder the true objective, potentially limiting the alignment capability of the LLMs. This issue is firstly studied by Gao et al. (2023), the results show that this issue may reduce the language diversity, generate unnatural language patterns to inflate rewards (Lazaridou and Baroni, 2020). Addressing this issue can be achieved by constraining the KL divergence term to the supervised fine-tuned reference model. But, calibrating the KL term requires careful hyperparameter tuning, which is computationally expensive with online policy optimization methods (Stiennon et al., 2020).

Recently, there are some works focus on solving the above issue based on the ensembles for the single or multiple objectives. For single objective, a representative way is to use the variance of the reward scores as the uncertainty, which are incorporated into the final reward scores to optimize the policy (Coste et al., 2023; Zhang et al., 2024a). Furthermore, with diverse ensembles (Zhai et al., 2023), the uncertainty quantification abilities can be enhanced with the diverse weights of the reward scores. For the multiple objectives, with different user preference, the problem can be solved by the Pareto reward model ensembles (Rame et al., 2024) and the prompt ensembles with

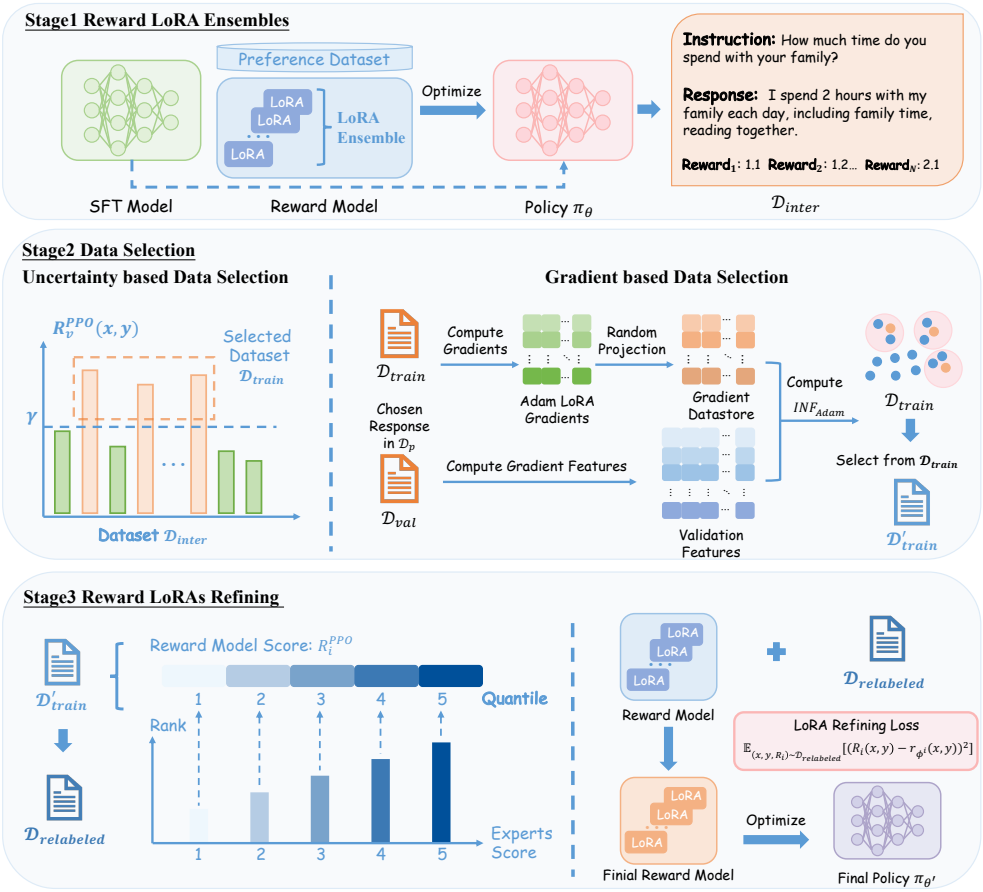



Figure 1: Overall training pipeline of our UGDA. The optimization is based on the Low-Rank Adaptation (LoRA) (Hu et al., 2021). We divide our pipeline into three stages, which are Reward LoRA Ensembles, Data Selection, Reward LoRAs Refining, respectively.

multiple policies (Jang et al., 2023). Our work focus on the single objective optimization. Thus, the main drawback of the aforementioned works lies in the fixed reward models. Due to this, they still encounter the problem of off-distribution. As the policy optimization continuously alters the distribution of the training dataset of the reward model, this poses a challenge for the reward model. The performance of the reward model may degrade, leading to a decline in the effectiveness of the policy that leverages the disparities between the estimated and "true" rewards (Gao et al., 2023). The above problem can be solved by collecting new preference data from the latest LLMs or the human annotator. But, gathering new data and annotating human preferences are costly, and will make the policy optimization be complicated. Hence, a new approach is required to get high quality data to update reward model efficiently.

In this paper, we propose a novel method named **U**ncertainty-**G**radient based **D**ata **A**ugmentation (**UGDA** for short) to achieve the policy-aware reward modeling. The training pipeline is shown in Figure 1. The core of our UGDA is to relabel the most uncertain and influential interaction data during policy optimization to further refine the reward model, which can mitigate the issue of reward model off-distribution. Specially, we construct the reward with uncertainty through the reward model ensembles. The mean and the variance of the reward scores are recorded, and the policy is trained by the mean score. Notably, the variance represents the uncertainty of the reward models. Then, with the interaction data during the policy training, to find the high quality data for reward model refining, we select the data by two aspects (*i.e.*, the uncertainty and the influence). The uncertainty based data selection is to select the samples with high uncertainty (*i.e.*, variance of the reward scores). Moreover, we observe that not all the high uncertainty samples are valuable for the reward model refining. Thus, we further select the data by the gradient based influence score, which can select the influential data for policy optimization. Leveraging the above selected data, we relabel the reward scores by introducing the experts, and the relabeled reward scores are projected to the distributions of

the corresponding proxy reward scores. The projected reward scores are used as the ground truth for the reward model refining, which can keep the reward model be on-distribution.

Our main contributions are summarized in the following:

- We design a novel method to collect the high quality data for reward model refining to solve the off-distribution reward overoptmization problem.
- We leverage the ensembles for the reward model uncertainty estimation and interaction data collection.
- We design an uncertainty-gradient based data selection for reward model refining, which can select the data with high uncertainty and be influential for the policy optimization.
- We conduct experiments to show the effectiveness of our UGDA, the results show that our UGDA can improve the performance compared with many state-of-the-art baselines.

## 2 Related Work

**Reward Uncertainty.** The challenge of an inaccurate proxy reward model in RLHF stems from its inability to fully represent human preferences, as highlighted in Ibarz et al. (2018); Ziegler et al. (2019); Stiennon et al. (2020). A common solution is adding a KL divergence penalty to keep the policy model close to the SFT model (Touvron et al., 2023; Yang et al., 2023; Ouyang et al., 2022). However, this can limit optimization and lead to overfitting (Azar et al., 2024; Gao et al., 2023). In recent, some works focus on modeling reward uncertainty using ensembles (Coste et al., 2023; Eisenstein et al., 2023; Lang et al., 2024), or using the last layer embedding of LLMs for uncertainty representation (Zhang et al., 2024b). But, the learned reward model still suffers from the off-distribution issue. Different from the above approaches, our method focuses on the on-distribution reward modeling data augmentation, aiming to refine the reward model's distribution in conjunction with policy optimization.

**Gradient based Data Selection.** Data selection is often considered a coreset selection problem (Phillips, 2017), which aims to identify a subset of training examples that can achieve performance comparable to training on the entire dataset (Toneva et al., 2018; Sener and Savarese, 2017; Coleman et al., 2019). These works concentrate on in-domain coreset selection. Several previous works utilize predefined concepts of valuable data (Gururangan et al., 2020; Chen et al., 2024) or n-gram features (Xie et al., 2023) to select pre-training examples. Xia et al. (2024) select data for the SFT by calculating the gradient features. Our method selects data by the gradient information, which is similar to (Mirzasoleiman et al., 2020; Killamsetty et al., 2021; Han et al., 2023; Xia et al., 2024). Different from the above works, we focus on select the on-distribution data generated by the optimization procedure of the policy, rather than select the offline training dataset.

## 3 Preliminaries

In this section, to better understand our method, we briefly introduce the RLHF for LLMs and the gradient-based data influence.

### 3.1 RLHF for LLMs

The RLHF for LLMs mainly contains three stages, the supervised fine-tuning, the reward modeling and the proximal policy optimization. To better understand our method, we mainly introduce the reward modeling and the proximal policy optimization in the following.

**Reward Modeling.** To approximate the human preference, RLHF involves the learning of the reward model, which leverages the annotated preference data. Given the preference dataset $\mathcal{D}_p$, reward model, initialized from the SFT policy with an added linear head, is designed to estimate the likelihood that a human would favor a specific completion. The reward model's loss function is designed by using the following calculation,

$$\mathcal{L}_R\left(r_\phi\right) = -\mathbb{E}_{(x,y_c,y_r)\sim\mathcal{D}_p}\left[\log\sigma\left(r_\phi\left(x,y_c\right) - r_\phi\left(x,y_r\right)\right)\right], \tag{1}$$

the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ is used here. It's applied to the human preference dataset $\mathcal{D}_p$, where $x$ is the input prompt for the language model. The human's selected response is indicated by $y_c$,

while the response they did not choose is $y_r$. The parameters of the reward model $r$ are represented by $\phi$.

**Proximal Policy Optimization.** Proximal Policy Optimization (PPO), as introduced in (Schulman et al., 2017), is an online reinforcement learning technique that leverages policy gradients. It iteratively refines the policy in small steps to optimize a specified reward function.

In the context of policy training with human feedback, PPO serves as a go-to method. Specifically, the reward for the policy being trained is determined by,

$$R^{\mathrm{PPO}}(x, y) = r_\phi(x, y) - \beta \mathbb{D}_{\mathrm{KL}} \left[ \pi_\theta(y \mid x) \| \pi^{\mathrm{SFT}}(y \mid x) \right]. \tag{2}$$

where $\beta$ is a hyperparameter controlling the strenth of the KL penalty, $\theta$ is the parameter of the policy $\pi_\theta$, $\pi^{\mathrm{SFT}}$ is the reference policy copied from the SFT policy.

### 3.2 Gradient based Data Influence

**Per-step influence.** At time step $t$, a model $\theta^t$ is trained using the loss function $\mathcal{L}(\cdot; \theta)$. The first-order Taylor expansion of this function for a validation instance $z'$ can be formulated as follows,

$$\mathcal{L}\left(z'; \theta^{t+1}\right) \approx \mathcal{L}\left(z'; \theta^t\right) + \left\langle \nabla \mathcal{L}\left(z'; \theta^t\right), \theta^{t+1} - \theta^t \right\rangle \tag{3}$$

To enhance the clarity of the explanation, we examine the model training process using Stochastic Gradient Descent (SGD) with a single-sample batch and a learning rate of $\eta_t$. At time step $t$, if $z$ denotes the training data, the SGD update formula is given by $\theta^{t+1} - \theta^t = -\eta_t \nabla \mathcal{L}(z; \theta^t)$. Following this, the Taylor expansion can be formulated as,

$$\mathcal{L}\left(z'; \theta^{t+1}\right) - \mathcal{L}\left(z'; \theta^t\right) \approx -\eta_t \left\langle \nabla \mathcal{L}\left(z; \theta^t\right), \nabla \mathcal{L}\left(z'; \theta^t\right) \right\rangle \tag{4}$$

**Trajectory influence.** To evaluate the cumulative effect of $z$ across the training process, one should sum its impact at each step where $z$ is involved. Since $z$ is applied once per epoch, it is suitable to express this cumulative impact as a summation across all epochs,

$$\mathrm{INF}_{\mathrm{SGD}}\left(z, z'\right) \triangleq \sum_{i=1}^{N} \bar{\eta}_i \left\langle \nabla \mathcal{L}\left(z'; \theta_i\right), \nabla \mathcal{L}\left(z; \theta_i\right) \right\rangle \tag{5}$$

where $\bar{\eta}_i$ denotes the learning rate for the $i$-th epoch out of the total $N$ epochs of training.

Measuring how each token affects the training process is really hard because it needs to figure out how changes in the model's settings affect each word's loss. So, we decide to pick data at the whole sentence level instead. Moreover, Large Language Models (LLMs) are typically trained employing the Adam optimizer, as described in (Kingma and Ba, 2014). The parameter updates at each step can be formulated by,

$$\begin{aligned} \theta^{t+1} - \theta^t &= -\eta_t \Gamma\left(z, \theta^t\right), \quad \Gamma\left(z, \theta^t\right) \triangleq \frac{m^{t+1}}{\sqrt{v^{t+1} + \epsilon}} \\ m^{t+1} &= \left(\beta_1 m^t + (1 - \beta_1) \nabla \mathcal{L}\left(z; \theta^t\right)\right) / \left(1 - \beta_1^t\right) \\ v^{t+1} &= \left(\beta_2 v^t + (1 - \beta_2) \nabla \mathcal{L}\left(z; \theta^t\right)^2\right) / \left(1 - \beta_2^t\right) \end{aligned} \tag{6}$$

Here, each calculation is done element-wise. $\beta_1$ and $\beta_2$ are hyperparameters for the first and second moments, respectively. There's also a small constant $\epsilon$ to prevent any math errors. To make Adam work well, we should pick $z$ so that it makes the product of $\nabla \mathcal{L}(z'; \theta^t)$ and $\Gamma(z, \theta^t)$ as big as possible. Then, we have the definition of Adam Influence.

**Definition 1** (Adam Influence (Xia et al., 2024)). *Suppose the model is trained for $N$ epochs, where $\bar{\eta}_i$ is the average learning rate in the $i$ th epoch and $\theta_i$ is the model checkpoint after the $i$-th epoch. We define the influence of a training sample $z$ on a validation sample $z'$ when training with Adam as,*

$$\mathrm{INF}_{\mathrm{Adam}}\left(z, z'\right) \triangleq \sum_{i=1}^{N} \bar{\eta}_i \frac{\left\langle \nabla \mathcal{L}\left(z'; \theta_i\right), \Gamma\left(z, \theta_i\right) \right\rangle}{\left\| \nabla \mathcal{L}\left(z'; \theta_i\right) \right\| \left\| \Gamma\left(z, \theta_i\right) \right\|} \tag{7}$$

# 4 METHODOLOGY

In a high level, our UGDA consists of three stages, Reward LoRA Ensembles, Data Selection and Reward LoRAs Refining (See Figure 1). Specially, *Reward LoRA Ensembles* is to construct the reward model uncetrainty by ensembles, and collect the interaction data of policy training. *Data Selection* aims to select the most uncertain and influential data as the high quality data for refining. *Reward LoRAs Refining* leverages the selected data to refine the reward model with the newly designed objective, and further retrain the policy. We present the details of each stage in the following.

## 4.1 REWARD LORA ENSEMBLES

Standard RLHF involves training a single reward model to estimate the true reward, which is subsequently utilized for policy optimization. However, various works in wider machine learning have demonstrated that training multiple estimators and integrating their outputs can measure uncertainty (Lakshminarayanan et al., 2017; Ovadia et al., 2019). Taking inspiration from this insight and motivated by (Coste et al., 2023), we propose to learn the LoRA ensembles of the reward model $\mathbb{R} = \left\{r_{\phi^1}, \ldots, r_{\phi^k}\right\}$ in the reward model training stage, where $r_{\phi^i}$ represents a LoRA of the reward model. In the process of policy optimization, we aggregate reward estimates from several reward models in the ensemble using mean optimization (Boyd and Vandenberghe, 2004), which simply takes the outputs of the different ensemble members,

$$R_m^{\text{PPO}}(x, y) = \frac{1}{k} \sum_{i=1}^{k} R_i^{\text{PPO}}(x, y), \quad \forall i \in \{1, \ldots, k\}. \tag{8}$$

where $R_i^{\text{PPO}}(x, y)$ is calculated by Equation (2) with the reward LoRA $r_{\phi^k}$. Additionally, we also collect the variance term $R_v^{\text{PPO}}(x, y) = \frac{1}{k} \sum_i \left(R_i^{\text{PPO}}(x, y) - R_m^{\text{PPO}}(x, y)\right)^2$ of the reward LoRA ensembles for uncertainty estimation. To generate the on-distribution samples of the policy, we leverage the reward LoRA ensembles in Equation (8) to optimize a policy $\pi_\theta$ by Equation (2), and collect all the interaction samples in $\mathcal{D}_{\text{inter}}$ in for reward LoRAs refining.

## 4.2 DATA SELECTION

**Uncertainty based Data Selection.** As the training step increases, the amount of the training samples is very large. Among the collected on-distribution samples, not all of them are necessary for the reward LoRAs refining. Then, we argue that the samples with high uncertainty (*i.e.*, high reward variance) are the potential samples which can improve the ability of the reward model.

Thus, we rank the collected samples $(x, y)$ by the reward variance $R_v^{\text{PPO}}(x, y)$, and set a threshold $\gamma$ for selecting samples with the highest uncertainty. Subsequently, we have,

$$\mathcal{D}_{\text{train}} = \left\{(x, y) | \operatorname{Rank}(R_v^{\text{PPO}}(x, y)) > \gamma, (x, y) \in \mathcal{D}_{\text{inter}}\right\}. \tag{9}$$

where $\mathcal{D}_{\text{train}}$ is the dataset used for the subsequent data selection, $\operatorname{Rank}(\cdot) \in [0, 1]$ represents the proportion of the ranked data. Moreover, in the main experiment, we set $\gamma = 0.5$.

**Gradient based Data Selection.** The main goal of the reward model is to optimize the policy. Therefore, it is essential to identify influential samples for policy optimization, as this can reduce the data sample size required for refining the reward model. Thus, after obtaining $\mathcal{D}_{\text{train}}$, we subsequently select the data by the gradient based influence function (Xia et al., 2024).

As defined in Definition 1, the gradient-based influence function is utilized on the trajectory with the Adam optimizer, leading to a notable expansion in the gradient's feature dimension, especially when fine-tuning LLMs. Consequently, aiming to diminish the dimensionality of features, we utilize a random projection method on the gradients obtained from LoRA. According to the Johnson-Lindenstrauss Lemma, as referenced in (Johnson and Naor, 2010), these projections typically maintain the inner products as defined in Definition 1. This preservation ensures that even the reduced-dimensionality gradient features remain valuable for the selection of datasets. The gradient based data selection process is depicted in Stage 2 of Figure 1, where it leverages the methodology introduced in Section 3.2 to effectively handle subtasks within the validation set. In the validation dataset $\mathcal{D}_{\text{val}}^{(j)}$,

**Algorithm 1** The training pipline of our UGDA.

**Require:** The SFT policy $\pi^{\text{SFT}}$, the human preference dataset $\mathcal{D}_p$.
**Ensure:** The trained policy $\pi_{\theta'}$
**REWARD LORA ENSEMBLES**
1: Training the reward LoRA ensembles with $\pi^{\text{SFT}}$ and $\mathcal{D}_p$ according to Equation (1).
2: Training the policy $\pi_\theta$ with the reward model $\mathbb{R}$ and $\pi^{\text{SFT}}$ according to Equation (2) and (8).
**UNCERTAINTY-GRADIENT BASED DATA SELECTION**
3: Selecting $\mathcal{D}_{\text{train}}$ from the collected data $\mathcal{D}_{\text{inter}}$ according to Equation (9).
4: Calculating the influence score $\text{INF}_{\text{Adam}}$ according to Equation (10) and (11).
5: Selecting $\mathcal{D}'_{\text{train}}$ from $\mathcal{D}_{\text{train}}$ according to the ranking of influence scores in Equation (12).
**REWARD LORAS REFINING**
6: Relabeling the reward of selected data $\mathcal{D}'_{\text{train}}$ with the introduced expert according to Equation (13).
7: Refining the reward LoRA ensembles according to Equation (14) and refining the policy $\pi_{\theta'}$ according to Equation (2).

corresponding to the $j$-th subtask, the average gradient feature is computed for each model checkpoint $\theta_1, \ldots, \theta_N$.

$$\bar{\nabla}\mathcal{L}\left(\mathcal{D}_{\text{val}}^{(j)}; \theta_i\right) = \frac{1}{\left|\mathcal{D}_{\text{val}}^{(j)}\right|} \sum_{(x,y)\in\mathcal{D}_{\text{val}}^{(j)}} \tilde{\nabla}\mathcal{L}\left((x,y);\theta_i\right), \tag{10}$$

For the given validation sample $(x', y')$ and model checkpoint $\theta_i$, with the $P$ dimensional gradient features, we can compute a $d$ dimensional projection of the LoRA gradient $\tilde{\nabla}\mathcal{L}\left((x',y');\theta_i\right) = \Pi^\top \hat{\nabla}\mathcal{L}\left((x',y');\theta_i\right)$, with each entry of $\Pi \in \mathbb{R}^{P\times d}$ drawn from a Rademacher distribution (*i.e.*, $\Pi_{ij} \sim \mathcal{U}(\{-1,1\})$ ). For training samples $(x,y)$, we compute $\tilde{\Gamma}((x,y),\cdot) = \Pi^\top \hat{\Gamma}((x,y),\cdot)$.

As in Definition 1, we aggregate the scores indicating the proximity of the sample to each validation subtask during training. The adaptation of Definition 1 can be formulated as follows,

$$\text{INF}_{\text{Adam}}\left((x,y), \mathcal{D}_{\text{val}}^{(j)}\right) = \sum_{i=1}^{N} \bar{\eta}_i \frac{\left\langle \bar{\nabla}\mathcal{L}\left(\mathcal{D}_{\text{val}}^{(j)}; \theta_i\right), \tilde{\Gamma}\left((x,y),\theta_i\right)\right\rangle}{\left\|\bar{\nabla}\mathcal{L}\left(\mathcal{D}_{\text{val}}^{(j)}; \theta_i\right)\right\| \left\|\tilde{\Gamma}\left((x,y),\theta_i\right)\right\|}. \tag{11}$$

We select training samples that can improve performance on any one of the validation samples. As described above, we compute the score for $(x,y)$ as the maximum across all subtasks: $\max_j \text{INF}_{\text{Adam}}\left((x,y), \mathcal{D}_{\text{val}}^{(j)}\right)$. After data selection in $\mathcal{D}_{\text{train}}$ , we use the selected subset $\mathcal{D}'_{\text{train}}$ to train the reward LoRA ensembles.

$$\mathcal{D}'_{\text{train}} = \left\{(x,y) | \operatorname{Rank}\left(\max_j \text{INF}_{\text{Adam}}\left((x,y), \mathcal{D}_{\text{val}}^{(j)}\right)\right) > \eta, (x,y) \in \mathcal{D}_{\text{train}}\right\}. \tag{12}$$

where $\eta$ is the threshold for the influence score selection, we set $\eta = 0.5$ in the main experiments. Thus, the size of $\mathcal{D}'_{\text{train}}$ is 25% of $\mathcal{D}_{\text{inter}}$.

### 4.3 REWARD LORAS REFINING

Table 1: Reward relabeling comparison of human labelers and GPT-4 by using the same prompt.

| **Score** | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Human Labelers | 24.9% | 21.3% | 19.8% | 17.6% | 16.4% |
| GPT-4 | 23.6% | 22.3% | 18.8% | 19.2% | 16.1% |
| **Similarity** (%) | 84.9% | 90.3% | 88.5% | 84.1% | 90.2% |

**Reward Relabeling.** Next, we need to use the dataset $\mathcal{D}'_{\text{train}}$ to refine the reward LoRA ensembles. We introduce the experts (*i.e.*, GPT-4 or Human) to relabel the rewards of these samples. Specially, we design the prompt with arbitrary scoring of 1 to 5, where 5 is the best possible score, the evaluation considers the correctness, helpfulness and harmfulness at the same time. The complete prompt is presented in Appendix D. Then, we use the labeled rewards to refine the reward LoRA ensembles. The reward score distributions of the reward LoRAs are different, thus we need to project the labeled reward

score of the data to the trained reward distribution. We use the quantile of 1 to 5 to correspond to the labeled score of 1 to 5,

$$R_i(x, y) = \text{Quantile\_5}(R_i^{\text{PPO}}(x, y), j) + \epsilon, \quad \forall i \in \{1, \ldots, k\}, \forall j \in \{1, \ldots, 5\}, \tag{13}$$

The ground truth reward for the sample $(x, y)$ of the reward LoRA $r_{\phi^i}(x, y)$ is denoted by $R_i(x, y)$. $\text{Quantile\_5}$ represents the data sorted in ascending order, with each step representing 20% of the data range from 20% to 100%. $\text{Quantile\_5}(p, q)$ denotes the $q$-th quantile of $p$, and $\epsilon \sim \mathcal{N}(0, 0.01)$ represents a small noise term. The relabeling results are presented in Table 1, where "**Similarity**" indicates the percentage of relabeled samples that match the results of GPT-4. The high degree of similarity in the results leads us to conveniently utilize the samples labeled by GPT-4 for more experiments.

**LoRAs Refining.** Then, we use the relabeled dataset $\mathcal{D}_{\text{relabeled}}$ to refine the reward LoRA ensembles. With the samples of projected ground truth rewards, we design a loss function for the reward model refining. Specifically, we use the Mean Square Error (MSE) loss for the reward regression fine-tuning.

$$\mathcal{L}'_R\left(r_{\phi^i}\right) = \mathbb{E}_{(x,y,R_i)\sim\mathcal{D}_{\text{relabeled}}}\left[\left(R_i\left(x, y\right) - r_{\phi^i}\left(x, y\right)\right)^2\right]. \tag{14}$$

where $r_{\phi^i} \sim \mathbb{R}$ represents each reward model, $R_i\left(x, y\right)$ is the relabeled reward for $i$-th reward LoRA. Then, with the refined reward LoRA ensembles, we can further fine-tune the policy $\pi_{\theta'}$ by Equation (2). To better understand our UGDA, we present the whole training pipeline in Algorithm 1.

## 5 Experiments

In this section, we conduct experiments to answer the following research questions.

- **RQ1**: Can our UGDA improve the performance of the trained policy compared with other baselines?
- **RQ2**: How is the influence of uncertainty-gradient based data selection in our UGDA?
- **RQ3**: How is the robustness of our UGDA with the noisy training data?
- **RQ4**: Can our UGDA improve the performance of reward model compared with other baselines?

Additionally, we also present some additional experimental results in Appendix C.

### 5.1 Experimental Setup

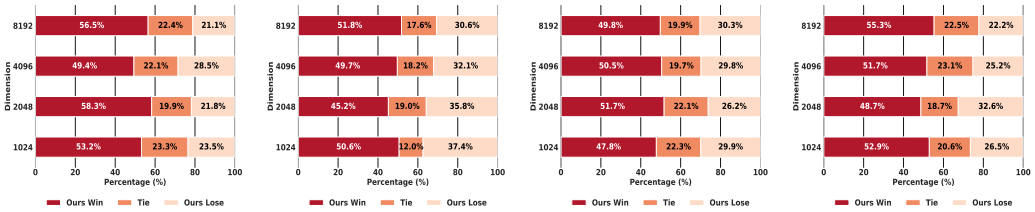
**Dataset and Models.** We use Anthropic's Helpful and Harmless (HH) dataset (Bai et al., 2022) as our experimental dataset. They provide a chosen response and a rejected response for each query based on human preferences. And for the validation dataset construction for the gradient based data selection, we use the instruction and chosen responses from the test sets of two subtasks (*i.e.*, helpful and harmless), the final influence scores are the mean of the two subtasks' influence scores. We conduct the experiments based on the Gemma-2B (Team et al., 2024) and Gemma-7B (Team et al., 2024) for the reward model, and the policy is conducted by the Gemma-7B. Additionally, to evaluate the performance of the reward model ensembles methods, we use a significantly larger model of Llama2-13B (Touvron et al., 2023) compared to our proxy reward models (with the largest being 7B), it is rational to apply it as the evaluation judge (Gao et al., 2023).

**Baselines.** We compare our method with Supervised Fine-Tuning (**SFT**) (Ouyang et al., 2022), Proximal Policy Optimization (**PPO**) (Ouyang et al., 2022), Lower Confidence Bound (**LCB**) (Zhang et al., 2024a), Uncertainty Weighted Optimization (**UWO**) (Coste et al., 2023), Reward LoRAs Retraining (**RLR**). Furthermore, to achieve the fair comparison, for the baselines, without loss of generality, we conduct experiments by randomly select 25% of the interaction data with GPT-4 annotator for the reward model refining.

**Metrics.** We conduct the evaluation from two perspectives (*i.e.*, the policy evaluation and the reward model evaluation). For the *policy evaluation*, we use two automatic metrics generated by the much larger reward model (*i.e.*, Llama2-13B): average of golden reward scores (Avg_Reward), variance of golden reward scores (Var_Reward). The responses generated by the different methods are also evaluated by the GPT-4 as a judge with the designed prompt, the evaluation prompt is shown in Appendix D. Also, we employed three popular and challenging benchmarks to evaluate the open-ended instruction following task: AlpacaEval (Li et al., 2023), Arena-Hard (Li et al., 2024), MT-Bench

Table 2: Larger reward model evaluation on HH dataset. All ensemble methods are implemented with three ensemble members, the best results and second best results are **bold** and underlined, respectively.

| Reward Model | Training Methods | Test Settings | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Helpful | | | | Harmless | | | |
| | | Avg_Reward (↑) | | Var_Reward (↓) | | Avg_Reward (↑) | | Var_Reward (↓) | |
| | | 0% | 25% | 0% | 25% | 0% | 25% | 0% | 25% |
| —- | SFT (Gemma-7B) | 0.62 | — | 0.18 | — | 0.58 | — | 0.16 | — |
| Gemma-2B | PPO | 0.62 | 0.64 | 0.16 | 0.18 | 0.59 | 0.60 | 0.18 | 0.19 |
| | LCB | 0.59 | 0.54 | 0.13 | 0.20 | 0.53 | 0.57 | 0.17 | 0.23 |
| | UWO | 0.56 | 0.51 | 0.14 | 0.19 | 0.58 | 0.58 | **0.13** | 0.17 |
| | RLR | — | 0.69 | — | 0.18 | — | 0.61 | — | 0.15 |
| | **UGDA** | — | **0.83** | — | **0.11** | — | **0.80** | — | 0.14 |
| Gemma-7B | PPO | 0.64 | 0.61 | 0.17 | 0.21 | 0.74 | 0.71 | 0.18 | 0.20 |
| | LCB | 0.70 | 0.73 | 0.15 | 0.19 | 0.73 | 0.75 | 0.11 | 0.16 |
| | UWO | 0.73 | 0.68 | 0.16 | 0.15 | 0.80 | 0.73 | **0.09** | 0.13 |
| | RLR | — | 0.79 | — | 0.14 | — | 0.71 | — | 0.14 |
| | **UGDA** | — | **0.91** | — | **0.12** | — | **0.88** | — | 0.10 |




(a) UGDA vs. PPO (b) UGDA vs. LCB (c) UGDA vs. UWO (d) UGDA vs. RLR

Figure 2: GPT-4 comparison on HH dataset. The presented results are of policy trained by using the Gemma-2B as the base reward model. The dimension represents the gradient projection dimension for gradient based data selection, the results across the dimensions are the hyperparameter analysis.

101 (Bai et al., 2024). For GPT-based evaluation, we employ GPT-4-turbo-2024-04-09 as the judge model to conduct pairwise comparisons for each preference optimization method. For the *reward model evaluation*, we show the accuracy of the reward models. The accuracy is computed by counting the percentage of the reward scores of good responses that are higher than bad responses. Additionally, we also conduct experiments on the commonly used benchmark RewardBench (Lambert et al., 2024), which evaluates the performance reward model from chat, reasoning, and safety perspectives.

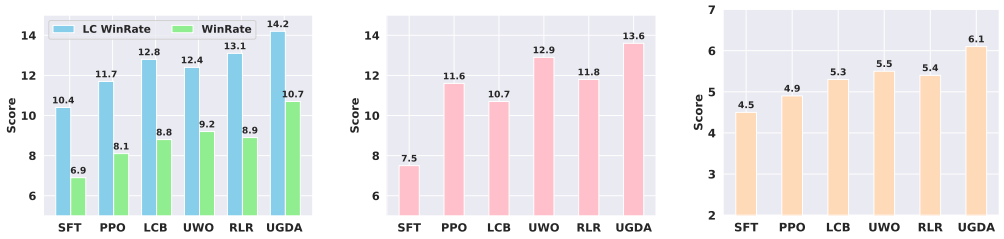
### 5.2 Implementation Details

We implement our UGDA and other baselines based on the LLaMA Factory (Zheng et al., 2024). Our RLHF pipelines consists several key stages. Initially, the learning of the proxy reward model are performed on Anthropic's HH dataset (Bai et al., 2022). Then, the policy optimization is conducted by the PPO algorithm (Ouyang et al., 2022). For the gradient based data selection, we use both the helpful and harmless validation dataset, the final influence score is the mean of the influence scores in the two subtasks. Further details are provided in Appendix B.

### 5.3 Policy Evaluation (RQ1, RQ2, RQ3)

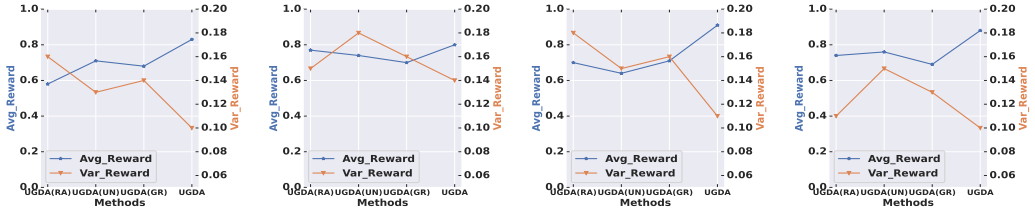
In this section, we conduct experiments to evaluate the performance of the learned policy.

**Overall Evaluation (RQ1).** For the larger reward model evaluation results in Table 2, we present the Average Reward (Avg_Reward) and Reward Variance (Var_Reward) of the trained policy using




(a) Alpaca Eval (b) Arena Hard (c) MTBench 101

Figure 3: Evaluation results on three instruction following benchmarks (*i.e.*, AlpacaEval, Arena-Hard, and MT-Bench 101).




(a) Helpful (2B) (b) Harmless (2B) (c) Helpful (7B) (d) Harmless (7B)

Figure 4: Ablation study of the uncertainty-gradient based data selection on HH dataset.

Gemma-2B and Gemma-7B as the base reward models. 0% represents the baselines without data augmentation. Additionally, the results are based on two test settings (*i.e.*, helpful and harmless). The policies trained using RLHF methods all significantly outperform the SFT policy on Avg_Reward. Since with sufficient training of the policy, a better trained reward model can help learning a better policy. Our UGDA achieves the best results across various metrics, demonstrating the effectiveness of data augmentation for reward modeling. The RLR achieves some of the best and second-best results, indicating that retraining the reward model on off-distribution samples can enhance the reward model's capability.

The GPT-4 evaluation results of the Gemma-2B-based reward model are depicted in Figure 2. Through meticulous parameter tuning and consideration of the specific model, the results demonstrate that UGDA outperforms responses from other baseline training methods, suggesting its potential in enhancing responses across various gradient projection dimensions of data selection. The instruction following benchmark results based on the Gemma-2B reward models are presented in Figure 3. The results are show that UGDA outperforms all the baselines across all benchmarks and settings.

**Ablation Study (RQ2).** Additionally, we conduct the ablation study to show the effectiveness of the uncertainty-gradient based data selection of our UGDA, and the results are presented in Figure 4. Specially, we make three variants (*i.e.*, UGDA (RA), UGDA (UN), UGDA (GR)), which represent random selection, only uncertainty based selection, only gradient based selection of the reward LoRAs refining data. From the results, we can see that, removing each part of the data selection will partially hurt the performance, which verify the effectiveness of the uncertainty-gradient based data selection.

**Robust Evaluation (RQ3).** To evaluate the robustness of our UGDA, we conduct experiments on a noisy dataset, where we randomly sampled 20% preference samples to change the position of each preference pair as the training data of the reward models. We use the larger reward model to evaluate the trained policy, the results are shown in Table 3, the GPT-4 comparison conducted on the Gemma-2B based reward models, the projection dimension of our UGDA is set to 8192, the results are shown in Figure 5. From the larger reward model evaluation results, we can see that the noisy data nearly decreases the model performance across all the baselines, but our UGDA suffers less change across most metrics. For the GPT evaluation, our UGDA still outperforms all the baseline methods, even getting better results compared with the pure data training results.

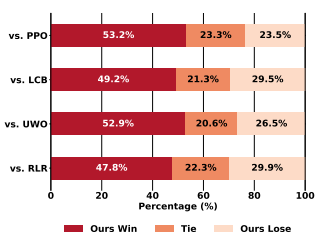



Figure 5: GPT-4 comparison for robustness. The policies are trained by using Gemma-2B as the reward model.

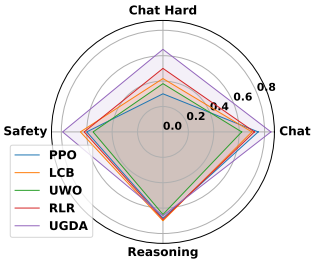



Figure 6: Evaluation results of Gemma-2B reward model on RewardBench.

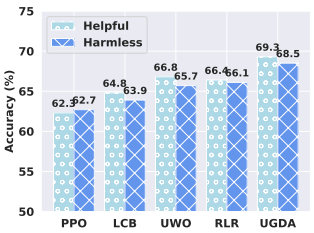



Figure 7: Gemma-2B reward model accuracy on the helpful and harmless test settings.

Table 3: Larger reward model evaluation for robustness on HH dataset. The best results and second best results are **bold** and underlined, respectively.

| Reward Model | Training Methods | Test Settings | | | |
|---|---|---|---|---|---|
| | | Helpful | | Harmless | |
| | | Avg_Reward (↑) | Var_Reward (↓) | Avg_Reward (↑) | Var_Reward (↓) |
| Gemma-2B | PPO | 0.51 | 0.22 | 0.48 | 0.21 |
| | LCB | 0.54 | 0.18 | 0.50 | 0.19 |
| | UWO | 0.60 | 0.18 | 0.52 | 0.16 |
| | RLR | 0.57 | 0.15 | 0.57 | 0.18 |
| | **UGDA** | **0.75** | **0.13** | **0.73** | **0.15** |
| Gemma-7B | PPO | 0.58 | 0.16 | 0.63 | 0.22 |
| | LCB | 0.61 | 0.14 | 0.60 | 0.16 |
| | UWO | 0.64 | 0.17 | 0.69 | **0.14** |
| | RLR | 0.67 | **0.12** | 0.72 | 0.17 |
| | **UGDA** | **0.81** | 0.15 | **0.83** | **0.14** |

### 5.4 Reward Model Evaluation (RQ4)

In this section, we conduct experiments to directly evaluate the learned reward model. The evaluation results based on the RewardBench are presented in Figure 6. Except for Reasoning, our UGDA can outperforms the reward models trained by the baselines on other perspectives. This may because that there are only few samples about the reasoning task in the filtered interaction data. The accuracy of the Gemma-2B based reward model is shown in Figure 7. The reward model trained by our UGDA can achieve 69.3% and 68.5% accuracy on the helpful and harmless testing sets, respectively. This indicates that our UGDA can enhance the response quality evaluation performance of the reward model.

## 6 Conclusion

In this paper, to solve the challenge of off-distribution reward modeling, we propose the UGDA, a novel method for policy-aware reward modeling. Specially, our UGDA is divided into three stages. (1) *Reward LoRA Ensembles* stage constructs the uncertainty of the reward model. (2) *Data Selection* stage selects the important on-distribution data by the uncertainty and gradient. (3) *Reward LoRAs Refining* stage introduces the experts to relabel the reward as the ground truth, and refines the reward LoRA ensembles. Then, we use the refined reward LoRA ensembles to train a new policy. Also, we conduct experiments to show the effectiveness of our UGDA, the results of the policy and reward model evaluations demonstrate the superiority over state-of-the-art baselines.

## REFERENCES


Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 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.

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.

Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, et al. Open problems and fundamental limitations of reinforcement learning from human feedback. *arXiv preprint arXiv:2307.15217*, 2023.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. PMLR, 2023.

Angeliki Lazaridou and Marco Baroni. Emergent multi-agent communication in the deep learning era. *arXiv preprint arXiv:2006.02419*, 2020.

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.

Thomas Coste, Usman Anwar, Robert Kirk, and David Krueger. Reward model ensembles help mitigate overoptimization. *arXiv preprint arXiv:2310.02743*, 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*, 2024a.

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.

Alexandre Rame, Guillaume Couairon, Corentin Dancette, Jean-Baptiste Gaya, Mustafa Shukor, Laure Soulier, and Matthieu Cord. Rewarded soups: towards pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards. *Advances in Neural Information Processing Systems*, 36, 2024.

Joel Jang, Seungone Kim, Bill Yuchen Lin, Yizhong Wang, Jack Hessel, Luke Zettlemoyer, Hannaneh Hajishirzi, Yejin Choi, and Prithviraj Ammanabrolu. Personalized soups: Personalized large language model alignment via post-hoc parameter merging. *arXiv preprint arXiv:2310.11564*, 2023.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in atari. *Advances in neural information processing systems*, 31, 2018.

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.

Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, et al. Baichuan 2: Open large-scale language models. *arXiv preprint arXiv:2309.10305*, 2023.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pages 4447–4455. PMLR, 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.

Hao Lang, Fei Huang, and Yongbin Li. Fine-tuning language models with reward learning on policy. *arXiv preprint arXiv:2403.19279*, 2024.

Xiaoying Zhang, Jean-Francois Ton, Wei Shen, Hongning Wang, and Yang Liu. Overcoming reward overoptimization via adversarial policy optimization with lightweight uncertainty estimation. *arXiv preprint arXiv:2403.05171*, 2024b.

Jeff M Phillips. Coresets and sketches. In *Handbook of discrete and computational geometry*, pages 1269–1288. Chapman and Hall/CRC, 2017.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. *arXiv preprint arXiv:1906.11829*, 2019.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*, 2020.

Mayee Chen, Nicholas Roberts, Kush Bhatia, Jue Wang, Ce Zhang, Frederic Sala, and Christopher Ré. Skill-it! a data-driven skills framework for understanding and training language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy S Liang. Data selection for language models via importance resampling. *Advances in Neural Information Processing Systems*, 36: 34201–34227, 2023.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: Selecting influential data for targeted instruction tuning. *arXiv preprint arXiv:2402.04333*, 2024.

Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning*, pages 6950–6960. PMLR, 2020.

Krishnateja Killamsetty, Sivasubramanian Durga, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. Grad-match: Gradient matching based data subset selection for efficient deep model training. In *International Conference on Machine Learning*, pages 5464–5474. PMLR, 2021.

Xiaochuang Han, Daniel Simig, Todor Mihaylov, Yulia Tsvetkov, Asli Celikyilmaz, and Tianlu Wang. Understanding in-context learning via supportive pretraining data. *arXiv preprint arXiv:2306.15091*, 2023.

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

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32, 2019.

Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

William B Johnson and Assaf Naor. The johnson–lindenstrauss lemma almost characterizes hilbert space, but not quite. *Discrete & Computational Geometry*, 43(3):542–553, 2010.

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*, 2022.

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.

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.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*, 2024.

Ge Bai, Jie Liu, Xingyuan Bu, Yancheng He, Jiaheng Liu, Zhanhui Zhou, Zhuoran Lin, Wenbo Su, Tiezheng Ge, Bo Zheng, et al. Mt-bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogues. *arXiv preprint arXiv:2402.14762*, 2024.

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.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*, 2024.