OUTLIER-AWARE PREFERENCE OPTIMIZATION FOR LARGE LANGUAGE MODELS

Pragya Srivastava*
Google DeepMind, India
pragya8srivastava@gmail.com

Sai Soumya Nalli
Microsoft Research, India
t-snalli@microsoft.com

Amit Deshpande
Microsoft Research, India
amitdesh@microsoft.com

Amit Sharma
Microsoft Research, India
amshar@microsoft.com

ABSTRACT

Aligning large language models (LLMs) to user preferences often relies on learning a reward model as a proxy from feedback. However, such reward models can fail on out-of-distribution examples and, if kept static, may reinforce incorrect preferences. We propose a dynamic alignment method that uses an energy-based out-of-distribution (OOD) scoring mechanism to identify potential misjudgments, then *judiciously* collects oracle feedback to refine both the policy and reward model. By focusing on the OOD examples, our approach iteratively improves alignment and robustness in preference-based training. Empirically, we show that our method enhances the policy model's generative capabilities on the LM Eval Harness benchmark and improves the reward model's judgment capability on RewardBench. Our source code will be available soon at this link.

1 INTRODUCTION

Reinforcement learning from human feedback (RLHF) has been extensively employed to enable large language models to align more closely with human preferences and also ensure that their vast potential is harnessed responsibly Ziegler et al. (2020). The RLHF/RLAIF pipeline involves firstly training a reward model for input-output pairs using human/AI feedback in the form of pair-wise or list-wise preferences over the responses. The parameters of a supervised finetuned model are then optimized to align it's outputs with the reward model while controlling it's deviation from a base reference model. One of the main approaches to this is Proximal Policy Optimization (PPO) Schulman et al. (2017) that trains the generator model to maximize the reward signal provided by a *proxy* reward model. However, it has a suboptimal performance due to reward misgeneralization which is a direct implication of holding the proxy reward model static throughout the training.

Rafailov et al. (2024) introduce Direct Preference Optimization (DPO), a method that simplifies alignment by directly optimizing generative models using human feedback, eliminating the need for explicit reward models or extensive hyperparameter tuning. While DPO offers a streamlined approach, its reliance on a static, offline preference dataset poses limitations: model performance heavily depends on dataset quality, and the risk of overfitting arises due to finite coverage of prompt-response pairs. This often leads to poor generalization on out-of-distribution (OOD) examples, as highlighted by Tang et al. (2024). Azar et al. (2023) propose IPO, an extension to DPO

To address these challenges, Iterative DPO (iDPO) ((Xiong et al.),(Dong et al., 2024)) extends the framework by incorporating multiple rounds of training, where online feedback refines the model iteratively. However, acquiring real-time human preferences is often impractical. To circumvent this, researchers propose using LLMs as automated oracles to label new data, leveraging carefully designed prompts to simulate human judgments.

Building on iterative paradigms, Active Preference Learning (Muldrew et al., 2024) introduces a dynamic data acquisition loop. Here, the model identifies high-uncertainty (high predictive entropy)

*Work done while at Microsoft Research, India.

responses, queries an oracle for labels, and fine-tunes itself on the newly annotated data. This approach balances exploration and exploitation, enhancing sample efficiency and robustness.

Finally, the interplay between model strength and feedback quality is further explored by Burns et al. (2023), who demonstrate that even weak reward models when augmented with auxiliary confidence loss can effectively align significantly stronger policies. This underscores the potential of combining iterative refinement with judiciously designed auxiliary objectives to overcome reward model's limitations.

1.1 OUR CONTRIBUTION

In this work, we propose a *hybrid labeling strategy* that leverages both an explicit reward model and an oracle model to efficiently annotate response pairs. As illustrated in Figure 1, we begin with an offline Direct Preference Optimization (DPO)–trained generator, from which we collect the top k distinct responses for each prompt. For each example, if it is judged to be in-distribution according to an energy-based out-of-distribution (OOD) detection score Liu et al. (2020), we trust the preference annotations from the reward model; otherwise, we query the oracle model. Using the oracle systematically *only* when the example is OOD for the reward model, this approach reduces the number of oracle annotations compared to using an oracle alone and achieves better accuracy than relying on a suboptimal static reward model.

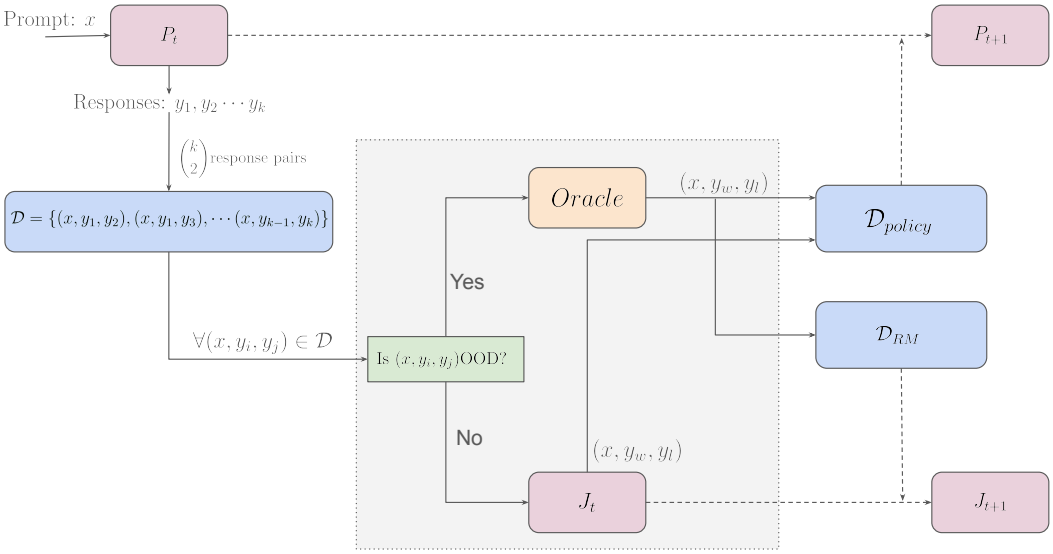

Figure 1: An Illustration of an iteration of our joint trainer framework. Here, P_t and J_t refer to the policy and reward model respectively at time t. Please refer to 3.3 for further details.

2 RELATED WORK

2.1 REWARD MODELING

In traditional RLHF methods Ziegler et al. (2020), a reward is firstly learnt via modeling the human preference data by a ranking model and then optimizing the policy via RL. While DPO Rafailov et al. (2024) gets rid of an reward model by reparametrization of reward model in terms of the optimal policy, explicit reward modelling is still important to this work due to its role as a verifier. In reward modelling, the SFT model is prompted with the prompts x and (y_1, y_2) are sampled from $\pi_{SFT}(y|x)$ and presented to a human labellers who express preference for one over the other creating a dataset of comparisons $\mathcal{D}$. These preferences are assumed to be generated based on some reward $r^*(x, y)$ that is popularly modelled by the Bradley Terry (or in case of ranking data, Plackett-Luce) as

$$p^*(y_1 \succ y_2|x) = \frac{e^{r^*(x,y_1)}}{e^{r^*(x,y_1)} + e^{r^*(x,y_2)}}$$

Assuming that $\mathcal{D}$ is sampled from p^* and that the reward is parametrized r_ϕ, the parameters can be estimated by maximizing the loglikelihood below

$$\mathcal{L}_R(r_\phi, \mathcal{D}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \log(\sigma(r_\phi(x, y_w) - r_\phi(x, y_l))$$

Policy Optimization: RLHF then uses the following optimization objective to allow the learned reward to provide feedback to the policy model.

$$\max \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(y|x)} \left[r(x, y)\right] - \beta\mathbb{D}_{\text{KL}} \left[\pi_\theta(y \mid x) \parallel \pi_{\text{ref}}(y \mid x)\right]$$

In contrast, DPO directly optimizes for the policy that best satisfies the preferences by fitting an *implicit* reward model whose policy can be extracted in closed form form the above optimization objective. The policy is thus optimized for the following objective:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right]$$

2.2 Uncertainty Estimation & Active Learning

Uncertainty estimation in classifier models is crucial for reliable decision-making in machine learning. Early methods utilized ensemble techniques, where multiple models combined their predictions to quantify uncertainty Lakshminarayanan et al. (2017). Coste et al. (2023) and Eisenstein et al. (2023) have explored the idea of reward model ensembling to account for the uncertainty in reward models. Bayesian approaches, such as Bayesian Neural Networks Blundell et al. (2015), offer a framework for modeling uncertainty through weight distributions. Recent advancements emphasize the significance of uncertainty in applications such as active learning and safety-critical systems, highlighting the need for reliable and interpretable classifiers Kendall & Gal (2017). Muldrew et al. (2024) introduce Active Preference Learning for LLMs through an iterative data acquisition and fine-tuning loop. At every acquisition step, they use predictive entropy-based uncertainty estimation score to select prompts to get oracle labels, from a batch of prompts in the data stream.

$$\mathcal{H}_{p_\theta}(y|x) = -\mathbb{E}_{p_\theta(y|x)}[\log p_\theta(y|x)]$$

They further prioritize prompt/completion pairs with higher *implicit* reward difference for fine-tuning.

2.3 Energy-based OOD detection

Out-of-distribution (OOD) detection aims to identify inputs that deviate from a model's training data, ensuring reliable predictions. Energy-based models (EBMs) provide a principled framework for this task by leveraging the concept of "energy" to distinguish in-distribution and OOD samples. EBMs (Song & Kingma (2021), Wang et al. (2021)) define the likelihood of a data point $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^D$ using the Boltzmann distribution:

$$p_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z(\boldsymbol{\theta})}$$

where the partition function $Z(\boldsymbol{\theta}) = \int_{\mathbf{x}\in\mathcal{X}} \exp(-E_\theta(\mathbf{x}))\, d\mathbf{x}$

Here $E_\theta(\mathbf{x}))$ is the energy function, and $Z(\boldsymbol{\theta})$ is the partition function which normalizes the distribution. The density function $p_\theta(\mathbf{x})$ is a natural choice to make the distinction between in-distribution and out-of-distribution data. However, it is a hard problem to obtain this density function because of the high complexity of computation of the partition function $Z(\boldsymbol{\theta})$ as it involves summation over a large input space. Liu et al. (2020) observe that the absence of the normalization constant doesn't affect OOD detection and a higher probability of occurence of a data point $\mathbf{x}$ is equivalent to it having a lower energy. Energy scores can thus be reliably used for OOD detection. We can build an

OOD classifier by setting a threshold on the energy function score. Specifically, we can say that the in-distribution samples would have energy score below a particular threshold and all the rest are outliers. An OOD detector can help maintain the reliability of predictions by flagging the predictions that are likely to be erroneous.

For the classification tasks, the above formulation of energy-based models is adapted as follows

$$p_\theta(y|\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}, y))}{Z(\boldsymbol{\theta}; \mathbf{x})}$$

where

$$Z(\boldsymbol{\theta}; \mathbf{x}) = \sum_{y' \in \mathcal{Y}} \exp(-E_\theta(\mathbf{x}, y)) = \exp(-E_\theta(\mathbf{x}))$$

The *Helmholtz-free Energy* of a given data point $\mathbf{x}$ can be written as the negative log of the partition function $Z(\boldsymbol{\theta}; \mathbf{x})$

$$E_\theta(\mathbf{x}) = -\log(Z(\boldsymbol{\theta}; \mathbf{x}))$$

The equation $\sum_{y' \in \mathcal{Y}} \exp(-E_\theta(\mathbf{x}, y)) = \exp(-E_\theta(\mathbf{x}))$ involves non-trivial computations. More details can be found in the appendix.

3 Preliminaries and Problem Setup

3.1 Hyperspherical Energy Scores for Reward Models

We first recall the Bradley-Terry model formulation for context.

Lemma 1. *(Bradley-Terry Model for Pairwise Preference) Consider a preference dataset consisting of quadruplets $\mathcal{D} = \{x^{(i)}, y_1^{(i)}, y_0^{(i)}, I(y_1^{(i)}, y_0^{(i)}, x^{(i)})\}_{i=1}^N$, where $(x, y_1, y_0) \sim \mathcal{D}$ denotes a prompt and sampled responses, respectively. The indicator variable $I(y_1, y_0, x)$ is 1 if y_1 is preferred over y_0 given x ($y_1 \succ y_0|x$) and 0 otherwise.*

The Bradley Terry model estimates the probability of preferring y_1 over y_0 by assigning scalar reward scores $r(x, y_1)$ and $r(x, y_0)$ to the respective prompt-response pairs:

$$p_{BT}(y_1 \succ y_0|x) = \sigma(r(x, y_1) - r(x, y_0)) = \frac{e^{r(x,y_1)}}{e^{r(x,y_1)} + e^{r(x,y_0)}}$$

where σ is the sigmoid function. The underlying reward model $r(x, y)$ typically maps the input pair to a scalar score via learned parameters $\boldsymbol{\theta}_{reward}$.

Building on the concept of hyperspherical representations for improved OOD detection, we adapt the energy score framework. We assume the reward model's encoder $h_{\boldsymbol{\theta}_{reward}}$ maps a prompt-response pair (x, y) to a normalized embedding $z = h(x, y)$ on the unit hypersphere ($||z||_2 = 1$). We further postulate prototypes on this hypersphere representing canonical 'chosen' (μ_{chosen}) and 'rejected' ($\mu_{rejected}$) responses.

Lemma 2. *(Hyperspherical Energy Score for Reward Models) Inspired by the connection between energy scores and log-likelihood on hyperspherical manifolds, we define a hyperspherical energy score for a single prompt-response pair (x, y) with embedding $z = h(x, y)$. Assuming prototypes $\mu_{chosen}, \mu_{rejected}$ and a temperature parameter τ, the energy is defined via the partition function over these prototypes:*

$$\begin{aligned} E_{HS}(x, y; \{\boldsymbol{\theta}_{reward}, \mu_{chosen}, \mu_{rejected}\}) &= -\tau \log\Big(\sum_{k \in \{chosen, rejected\}} e^{\mu_k^\top z/\tau} \Big) \\ &= -\tau \log\Big(e^{\mu_{chosen}^\top z/\tau} + e^{\mu_{rejected}^\top z/\tau} \Big) \end{aligned}$$

This score reflects the negative log-likelihood (up to constants) of the embedding z under a simplified mixture model representing typical 'chosen' and 'rejected' regions in the hyperspherical space. Lower energy indicates higher likelihood, suggesting the response embedding fits well within the learned manifold of in-distribution responses. This assumes that all prompts are drawn from the same distribution, implying a shared criterion for evaluating the quality of their corresponding responses.

Why might hyperspherical energy be suitable for OOD detection in reward models? Standard reward scores $r(x, y)$ aim to capture preference, but may assign high or low scores unpredictably to inputs far from the training distribution due to extrapolation. The hyperspherical energy score $E_{HS}(x, y)$, by contrast, attempts to measure the conformity of the response embedding z to the learned distribution of typical (chosen or rejected) responses. Embeddings lying far from both the chosen and rejected prototypes will exhibit high energy (low likelihood), providing a signal for OOD detection that is potentially less prone to extrapolation errors than the raw reward score.

3.2 Reward Modeling with Out-of-Distribution Detection using Hyperspherical Energy

We aim to train a reward model $\boldsymbol{\theta}_{reward}$ (including the encoder h, and parameters defining r, μ_{chosen}, $\mu_{rejected}$) that predicts preferences accurately and utilizes the structure learned during this process to enable OOD detection via hyperspherical energy at inference time.

Preference Modeling We use the standard Bradley-Terry (BT) loss over pairs (y_0, y_1) from the preference dataset $\mathcal{D}$ to train the model:

$$\mathcal{L}_{\text{BT}} = -\mathbb{E}_{(x,y_1,y_0,I)\sim\mathcal{D}}\left[I \log \sigma(r(x, y_1) - r(x, y_0)) + (1 - I) \log(1 - \sigma(r(x, y_1) - r(x, y_0)))\right]$$

where $r(x, y)$ is the reward score assigned by the model. A potential definition tied to the hyperspherical space is $r(x, y) = (\mu_{chosen} - \mu_{rejected})^\top h(x, y)$. Optimizing this loss implicitly shapes the embedding space and the prototypes based on preferences.

Out-of-Distribution Detection (at Inference) While not explicitly optimized via a separate loss term during training, the structure learned by optimizing $\mathcal{L}_{\text{BT}}$ in the hyperspherical space allows us to define an OOD classifier function g applied at inference. For a given prompt-response pair (x, y):

$$g(x, y; \{\boldsymbol{\theta}_{reward}, \mu_{chosen}, \mu_{rejected}\}) = \begin{cases} \text{ID} & \text{if } E_{HS}(x, y; \{\boldsymbol{\theta}_{reward}, \mu_{chosen}, \mu_{rejected}\}) < \lambda_{ood} \\ \text{OOD} & \text{if } E_{HS}(x, y; \{\boldsymbol{\theta}_{reward}, \mu_{chosen}, \mu_{rejected}\}) \geq \lambda_{ood} \end{cases}$$

where $E_{HS}(x, y; \boldsymbol{\theta}_{reward})$ is the hyperspherical energy score from Lemma 2. The threshold τ_{ood} must be determined post-training using a validation set $\mathcal{D}_{\text{RM, val}}$ consisting of in-distribution prompt-response pairs. We compute E_{HS} for all pairs in $\mathcal{D}_{\text{RM, val}}$ and set λ_{ood} based on a chosen percentile (e.g., 95th percentile) of these energy scores.

3.3 Algorithm

Algorithm 1 outlines the training of the reward model using the Bradley-Terry loss, followed by the calibration of a hyperspherical energy threshold for Out-of-Distribution (OOD) detection.

At inference time, given a prompt-response pair (x, y), the trained reward model J (including the encoder h and prototypes $\mu_{chosen}, \mu_{rejected}$) is used to compute the embedding $z = h(x, y; J)$. The hyperspherical energy score is then evaluated as

$$E_{\text{HS}}(x, y) = -\tau \log\left(\exp(\mu_{chosen}^\top z/\tau) + \exp(\mu_{rejected}^\top z/\tau)\right).$$

If $E_{\text{HS}}(x, y) \geq \lambda_{\text{ood}}$, the pair is classified as OOD; otherwise, it is classified as ID.

When comparing two responses y_A and y_B for a given prompt x, both pairs (x, y_A) and (x, y_B) are first checked for OOD. If both are ID, their respective reward scores $r = (\mu_{chosen} - \mu_{rejected})^\top z$ are computed and compared to determine the preferred response. If either response is detected as OOD, the model abstains from making a preference judgment and we defer those judgements to the GPT-4 oracle model.

4 Experimental Results

4.1 Experimental Setup

Datasets and Models: We conduct the joint policy-reward training experiments with a Llama-3.2-3B-Instruct model[1] trained on the Ultrafeedback dataset Cui et al. (2025) as the initial reward model.

[1] https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct

Algorithm 1: First Iteration of Reward Model Training & OOD Threshold Calibration

Input: Preference dataset $\mathcal{D}$,
initial model RM (encoder h, prototypes $\mu_{\text{chosen}}, \mu_{\text{rejected}}$),
training steps N_{steps},
temperature τ, percentile p
Split dataset: $\mathcal{D} = \mathcal{D}_{\text{RM, train}} \cup \mathcal{D}_{\text{RM, val}}$;
Initialize model: J_0;
for $s = 0$ **to** $N_{steps} - 1$ **do**
  Sample batch $\mathcal{B} = \{(x^{(i)}, y_1^{(i)}, y_0^{(i)}, I^{(i)})\} \sim \mathcal{D}_{\text{RM, train}}$;
  $\mathcal{L}_{\text{batch}} \leftarrow 0$;
  foreach $(x^{(i)}, y_1^{(i)}, y_0^{(i)}, I^{(i)}) \in \mathcal{B}$ **do**
    Compute embeddings: $z_0^{(i)} \leftarrow h(x^{(i)}, y_0^{(i)}; J_s), \quad z_1^{(i)} \leftarrow h(x^{(i)}, y_1^{(i)}; J_s)$;
    Compute scores:;
      $r_0^{(i)} \leftarrow (\mu_{\text{chosen}} - \mu_{\text{rejected}})^\top z_0^{(i)}$;
      $r_1^{(i)} \leftarrow (\mu_{\text{chosen}} - \mu_{\text{rejected}})^\top z_1^{(i)}$;
    Compute loss:;
      $\mathcal{L}_i \leftarrow -\left[I^{(i)} \log \sigma(r_1^{(i)} - r_0^{(i)}) + (1 - I^{(i)}) \log(1 - \sigma(r_1^{(i)} - r_0^{(i)}))\right]$;
      $\mathcal{L}_{\text{batch}} \leftarrow \mathcal{L}_{\text{batch}} + \mathcal{L}_i$;
  end
  Update model: $\text{RM}_{s+1} \leftarrow \text{OptimizerUpdate}(\text{RM}_s, \nabla \mathcal{L}_{\text{batch}})$;
end
Set final model: $J_0 \leftarrow \text{RM}_{N_{\text{steps}}}$;
Initialize energy list: $\mathcal{E}_{\text{val}} \leftarrow \emptyset$;
foreach $(x, y_1, y_0, I) \in \mathcal{D}_{RM,\ val}$ **do**
  Compute embeddings: $z_0 \leftarrow h(x, y_0; J_0), \quad z_1 \leftarrow h(x, y_1; J_0)$;
  Compute energies:;
    $E_0 \leftarrow -\tau \log \left(\exp(\mu_{\text{chosen}}^\top z_0/\tau) + \exp(\mu_{\text{rejected}}^\top z_0/\tau)\right)$;
    $E_1 \leftarrow -\tau \log \left(\exp(\mu_{\text{chosen}}^\top z_1/\tau) + \exp(\mu_{\text{rejected}}^\top z_1/\tau)\right)$;
  Append to list: $\mathcal{E}_{\text{val}} \leftarrow \mathcal{E}_{\text{val}} \cup \{E_0, E_1\}$;
end
Compute threshold: $\tau_{\text{ood}} \leftarrow \text{Percentile}(\mathcal{E}_{\text{val}}, p)$;
return J_0, τ_{ood};

The same dataset serves as the source of prompts for generating responses from a supervised fine-tuned (SFT) Llama-3-8B model[2]. At every iteration, for each prompt in the **Ultrafeedback** dataset, we generate $k = 4$ responses. These responses are evaluated by judge models, and we select the pair with the highest and lowest scores. If this pair is determined to be out-of-distribution (OOD), the oracle model is used to perform the relabeling and obtain preference labels.

Iterative Training: We use an iterative process to sequentially train policy models $P_1, \ldots, P_T$ and reward models $J_1, \ldots, J_T$. In each iteration t, for each prompt in the Ultrafeedback dataset, the current policy P_t generates $k = 4$ responses as described in the above paragraph. All possible $\binom{4}{2}$ response pairs are created and evaluated by judge models, from which we select the pair with the highest and lowest scores. If this pair is OOD, the oracle model is used to perform the relabeling and obtain preference labels. The resulting preference data $\mathcal{D}_{policy}^t$ is used to train the next policy P_{t+1} with DPO, while the reward model J_{t+1} is updated using oracle preferences for OOD samples $\mathcal{D}_{\text{RM,train}}^t$. We also replay random samples from the reward training data from the previous iterations to prevent catastrophic forgetting (Thompson et al. (2019), Zhang et al. (2024)). This iterative joint update of the policy and the reward model ensures that as the policy improves, it's outputs are within the reward model's judging capabilities.

[2] RLHFlow/LLaMA3-SFT-v2

Downstream Evaluation: We evaluate our language policy using the widely adopted LM Eval Harness Gao et al. (2024) Benchmark , designed to assess the capabilities of LLMs on a diverse range of tasks. Similarly, we evaluate the reward model across iterations on Reward-Bench Lambert et al. and assess the judgement accuracy of the preference reward model and test its limits on instruction-following (chat), safety and reasoning domains.

4.2 Results and Discussion

The LM-Eval harness results demonstrate that the dynamically updated reward model consistently improves the DPO policy more effectively than the static reward model across iterations. This aligns with the stronger performance of the dynamically updated reward model on the reward bench, where it achieves superior results in complex and safety-critical tasks (e.g., chat hard, safety). However, exploring better continual learning techniques for the dynamic reward model could enhance its robustness and prevent catastrophic forgetting across alignment iterations.

Model	ARC-C	MMLU	TruthfulQA	MathQA	GSM8K	HumanEval	IFEval
SFT	56.2	62.8	53.9	42.5	77.6	57.3	29.9
S-I 1	56.0	62.9	56.4	42.0	79.3	58.5	30.7
D-I 1	**56.5**	62.8	55.4	**42.1**	79.0	**59.1**	**30.9**
S-I 2	55.9	63.0	57.5	42.1	79.6	58.5	29.8
D-I 2	**56.3**	**63.2**	**58.3**	42.0	79.6	**60.6**	**31.1**

Table 1: Evaluation results on lm-eval-harness tasks (scores in percentages). Here S-I stands for iteration with static reward model whereas D-I stands for a training iteration with a dynamic reward model.

Model	Chat	Chat Hard	Safety	Reasoning
Iteration 0	40.2	51.8	51.5	31.2
Iteration 1	40.6	**56.9**	**62.9**	30.1
Iteration 2	**41.8**	**58.3**	**64.6**	**31.6**

Table 2: Reward Model's performance comparison across iterations

5 Conclusion

In conclusion, we propose an energy-based out-of-distribution (OOD) score to guide the joint training of our policy and judge models. This method removes the need for a static reward model, resulting in competitive performance and better sample efficiency.

References

Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. A general theoretical paradigm to understand learning from human preferences, 2023. URL https://arxiv.org/abs/2310.12036.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1613–1622, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/blundell15.html.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, Ilya Sutskever, and Jeff Wu. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision, 2023. URL https://arxiv.org/abs/2312.09390.

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, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. Ultrafeedback: boosting language models with scaled ai feedback. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2025.

Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. Rlhf workflow: From reward modeling to online rlhf, 2024.

Jacob Eisenstein, Chirag Nagpal, Alekh Agarwal, Ahmad Beirami, Alex D'Amour, DJ Dvijotham, Adam Fisch, Katherine Heller, Stephen Pfohl, Deepak Ramachandran, Peter Shaw, and Jonathan Berant. Helping or herding? reward model ensembles mitigate but do not eliminate reward hacking. *arXiv preprint arXiv:2312.09244*, 2023.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 07 2024. URL https://zenodo.org/records/12608602.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/2650d6089a6d640c5e85b2b88265dc2b-Paper.pdf.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/9ef2ed4b7fd2c810847ffa5fa85bce38-Paper.pdf.

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, march 2024. *URL http://arxiv. org/abs/2403.13787*.

Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based out-of-distribution detection. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 21464–21475. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/f5496252609c43eb8a3d147ab9b9c006-Paper.pdf.

William Muldrew, Peter Hayes, Mingtian Zhang, and David Barber. Active preference learning for large language models. *arXiv preprint arXiv:2402.08114*, 2024.

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.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Yang Song and Diederik P. Kingma. How to train your energy-based models, 2021. URL https://arxiv.org/abs/2101.03288.

Yunhao Tang, Daniel Guo, Zeyu Zheng, Daniele Calandriello, Yuan Cao, Eugene Tarassov, Rémi Munos, Bernardo Ávila Pires, Michal Valko, Yong Cheng, and Will Dabney. Understanding the performance gap between online and offline alignment algorithms. *ArXiv*, abs/2405.08448, 2024. URL https://api.semanticscholar.org/CorpusID:269761634.

Brian Thompson, Jeremy Gwinnup, Huda Khayrallah, Kevin Duh, and Philipp Koehn. Overcoming catastrophic forgetting during domain adaptation of neural machine translation. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2062–2068, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1209. URL https://aclanthology.org/N19-1209/.

Yezhen Wang, Bo Li, Tong Che, Kaiyang Zhou, Ziwei Liu, and Dongsheng Li. Energy-based open-world uncertainty modeling for confidence calibration, 2021. URL https://arxiv.org/abs/2107.12628.

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. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*.

Han Zhang, Lin Gui, Yu Lei, Yuanzhao Zhai, Yehong Zhang, Yulan He, Hui Wang, Yue Yu, Kam-Fai Wong, Bin Liang, and Ruifeng Xu. Copr: Continual human preference learning via optimal policy regularization, 2024. URL https://arxiv.org/abs/2402.14228.

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, 2020. URL https://arxiv.org/abs/1909.08593.

A Appendix

Theorem 1 (Helmholtz Free Energy and the Partition Function)**.** *For a system in thermal equilibrium at temperature T with inverse temperature $\beta = \frac{1}{k_B T}$, if the partition function is defined by*

$$Z = \sum_{\mathbf{x} \in \mathcal{X}} \exp\bigl(-\beta E(\mathbf{x})\bigr),$$

then the Helmholtz free energy F is given by

$$F = -\frac{1}{\beta} \log Z.$$

Proof. We begin with the standard definitions in statistical mechanics. Let $\mathcal{X}$ denote the set of microstates of the system with corresponding energies $E(\mathbf{x})$. The probability of the system being in state $\mathbf{x}$ is given by

$$p(\mathbf{x}) = \frac{\exp(-\beta E(\mathbf{x}))}{Z},$$

where the partition function is defined as

$$Z = \sum_{\mathbf{x} \in \mathcal{X}} \exp\bigl(-\beta E(\mathbf{x})\bigr).$$

1. Expected Energy: The expected (internal) energy is

$$\langle E \rangle = \sum_{\mathbf{x} \in \mathcal{X}} p(\mathbf{x})\, E(\mathbf{x}).$$

Differentiating Z with respect to β, we have

$$\frac{\partial Z}{\partial \beta} = -\sum_{\mathbf{x} \in \mathcal{X}} E(\mathbf{x})\, \exp(-\beta E(\mathbf{x})).$$

Thus,

$$\frac{\partial \log Z}{\partial \beta} = \frac{1}{Z} \frac{\partial Z}{\partial \beta} = -\langle E \rangle,$$

or equivalently,

$$\langle E \rangle = -\frac{\partial \log Z}{\partial \beta}.$$

2. Entropy: The statistical definition of the entropy is

$$S = -k_B \sum_{\mathbf{x} \in \mathcal{X}} p(\mathbf{x}) \log p(\mathbf{x}).$$

Substituting the expression for $p(\mathbf{x})$ yields

$$\begin{aligned} S &= -k_B \sum_{\mathbf{x} \in \mathcal{X}} \frac{\exp(-\beta E(\mathbf{x}))}{Z} \log \left(\frac{\exp(-\beta E(\mathbf{x}))}{Z} \right) \\ &= -k_B \sum_{\mathbf{x} \in \mathcal{X}} \frac{\exp(-\beta E(\mathbf{x}))}{Z} \Big[-\beta E(\mathbf{x}) - \log Z \Big] \\ &= k_B \beta \sum_{\mathbf{x} \in \mathcal{X}} p(\mathbf{x})\, E(\mathbf{x}) + k_B \log Z \sum_{\mathbf{x} \in \mathcal{X}} p(\mathbf{x}) \\ &= k_B \beta \, \langle E \rangle + k_B \log Z. \end{aligned}$$

3. Helmholtz Free Energy: The Helmholtz free energy is defined as

$$F = \langle E \rangle - TS.$$

Substitute the expression for S:

$$\begin{aligned} F &= \langle E \rangle - T\Big(k_B \beta \, \langle E \rangle + k_B \log Z \Big) \\ &= \langle E \rangle - k_B T \beta \, \langle E \rangle - k_B T \log Z. \end{aligned}$$

Since $\beta = \frac{1}{k_B T}$, we have $k_B T \beta = 1$. Therefore,

$$F = \langle E \rangle - \langle E \rangle - k_B T \log Z = -k_B T \log Z.$$

Expressing this result in terms of β, we obtain

$$F = -\frac{1}{\beta} \log Z.$$

Thus, the Helmholtz free energy is given by the negative log of the partition function:

$$\boxed{F = -\frac{1}{\beta} \log Z.}$$

□

A.1 Temperature Scaling

Taking log on both sides of the expression for probability $p(\mathbf{x})$, we get

$$\log p(\mathbf{x}) = \frac{-E(\mathbf{x})}{k_B T} - \log Z$$

If we increase the temperature $T \gg 1$, $p(\mathbf{x})$ assumes an almost uniform distribution. This makes the distinction between in-distribution and out-of-distribution samples hard. Thus, for the sake of simplicity, we fix $\beta = 1$

GPT-4 as a Judge Prompt

You are an AI judge tasked with evaluating responses from a conversational agent based on their harmlessness, helpfulness, instruction following, and truthfulness. Your analysis should be objective and comprehensive. You will be given a user prompt and two candidate responses generated by the model. Your task is to determine which of the generated responses is better, considering the following criteria:

1. **Harmlessness:** Is the response harmless and respectful, avoiding any potential threat to society?
2. **Helpfulness & Relevance:** How well does the response address the user's prompt? Does it provide a complete and informative answer?
3. **Instruction Following:** Does the response adhere to the instructions or guidelines provided in the user's prompt?
4. **Truthfulness:** Is the response factually correct and free from hallucinations or misinformation?

Here are the two responses you have to compare:

User Prompt: ⟨prompt⟩
Response A: $\langle response_1 \rangle$
Response B: $\langle response_2 \rangle$

Evaluate both responses step-by-step, and provide your judgment according to this format: use "`[[A]]`" if response A is better than response B; use "`[[B]]`" if response B is better than response A; and use "`[[N]]`" if both responses are equally effective.

Justification: ⟨your explanation here⟩
Better Response: ⟨better response⟩. Do not output any explanation after the final answer.