SYMMETRIC REINFORCEMENT LEARNING LOSS FOR ROBUST LEARNING ON DIVERSE TASKS AND MODEL SCALES

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) training is inherently unstable due to factors such as moving targets and high gradient variance. Reinforcement Learning from Human Feedback (RLHF) and Reinforcement Learning from AI Feedback (RLAIF) introduce additional challenges. For instance, diverse preferences complicate the alignment process, and prediction errors in a trained reward model can become more severe as the LLM generates unseen outputs. These RL challenges create confusion about whether the probability of an action for a given state should be increased or decreased, similar to the noise in labels for classification tasks. In this work, we enhance the stability of the RL training procedure by adapting reverse cross-entropy (RCE) from supervised learning for noisy data to define a symmetric RL loss. We demonstrate performance improvements across various tasks and scales. We conduct experiments in discrete action tasks (Atari games) and continuous action space tasks (MuJoCo benchmark and Box2D) using Symmetric A2C (SA2C) and Symmetric PPO (SPPO), with and without added noise. Notably, SPPO shows strong performance across different hyperparameters. Furthermore, we validate the benefits of the symmetric RL loss in the RLHF framework using PPO for natural language processing tasks, demonstrating improved performance in tasks such as IMDB positive sentiment and TL;DR summarization.

1 INTRODUCTION

Recent advancements in Large Language Models (LLMs) have shown impressive performance across various natural language processing tasks (Chung et al., 2022; Wei et al., 2023), robot control (Huang et al., 2022; Driess et al., 2023), and healthcare (Lee et al., 2023c; Huang et al., 2020). However, as these LLMs are typically trained to predict the next word in a provided dataset, they require post-training processing to make them useful for particular tasks. Reinforcement Learning from Human Feedback (RLHF) trains LLMs to generate responses aligned with user preferences through human feedback. Additionally, Reinforcement Learning from AI Feedback (RLAIF), which leverages feedback from well-trained AI models, has also been employed (Lee et al., 2023a; Bai et al., 2022). Thus, adapting fundamental Reinforcement Learning (RL) algorithms such as REINFORCE (Williams, 1992), A2C (Mnih et al., 2016), and PPO (Schulman et al., 2017) to suit the fine-tuning of LLMs for LLM tasks is an area of active interest (Ahmadian et al., 2024; Ouyang et al., 2022; Rafailov et al., 2023).

RL methods (Sutton et al., 2000; Sutton & Barto, 2018a) have lead to substantial breakthroughs in tasks such as robot control and game playing. Still, they entail learning instability compared to supervised learning due to factors such as moving targets, high-gradient variance, and training value functions. The RL literature has proposed various methods to make the RL learning process more robust, such as preventing overestimation with Double DQN (van Hasselt et al., 2015), reducing variance with Generalized Advantage Estimation (GAE) (Schulman et al., 2018), updates within the trust region (Schulman et al., 2015; 2017), and encouraging diverse behavior with Soft Actor-Critic (SAC) (Haarnoja et al., 2018). In addition to the methods devised specifically for RL problems, RL literature has also adopted supervised learning techniques to make the learning process more robust. For example, ensembles have been used for more accurate value function prediction, while Layer

Normalization and Batch Normalization have been employed to constrain predictions for out-of-distribution samples, thereby mitigating the overestimation and extrapolation.

RLHF (Ouyang et al., 2022; Lee et al., 2023b) and RLAIF Lee et al. (2023a); Bai et al. (2022); Byun et al. (2024) potentially introduce additional training challenges. For example, these algorithms often receive feedback from multiple sources (human or AI models) to align LLMs, and each feedback provider may have different preferences, meaning a sample considered preferable by one provider could be deemed undesirable by another (Ethayarajh et al., 2024; Chakraborty et al., 2024). In addition, RLHF and RLAIF often leverage a trained reward model to provide feedback on samples generated by the LLM. This indirection raises the question: *does the learned reward model provide the correct reward?* The reward model has prediction errors itself (See Figure 1), but as the LLM is trained with RL, its outputs deviate from the reward model's training dataset, introducing more error in the reward model's predictions for out-of-distribution samples.

The challenges associated with RL, RLHF, and RLAIF, as mentioned above, can introduce confusion when calculating advantage values in RL algorithms like A2C and PPO. Specifically, an action that should have a positive advantage value may have a negative sign in the next update, depending on which samples (states, actions) are generated and how the batch is composed during advantage normalization. The sign of the advantage determines whether the probability of a corresponding action for a given state increases or decreases in policy gradient algorithms. If the advantages are predicted incorrectly, this can lead to learning in the opposite direction. We hypothesize that these difficulties are similar to noisy classification tasks in supervised learning, where some labels are incorrect.

In this paper, we leverage a technique developed for classification tasks with noisy labels, employing a robust loss function to enhance the learning procedures of A2C and PPO. We define a symmetric RL loss, whose fundamental mechanism aligns with the robust loss function used in supervised learning (Wang et al., 2019), to improve the robustness of RL procedure for A2C and PPO (See Section 4.3). We apply this symmetric RL loss to A2C and PPO, naming them Symmetric A2C (SA2C) and Symmetric PPO (SPPO), and evaluate their performance across various tasks and model scales.

First, we assess the performance gains of SA2C and SPPO on Atari games (Mnih et al., 2016), which have discrete action spaces, as well as on the MuJoCo benchmark (Todorov et al., 2012) and Box2D (Catto, 2011) environments, which have continuous action spaces. For these control tasks, we introduce a noisy reward variant, hypothesizing that it will increase confusion in advantage prediction to better evaluate our method. Additionally, we test our method on RLHF tasks using LLMs, such as IMDB positive sentiment analysis (Maas et al., 2011) and TL;DR summarization (Völske et al., 2017). The IMDB task involves generating positive sentiment for a given context and TL;DR is a summarization task where an LLM is required to summarize content.

SA2C and SPPO demonstrate better performance improvements across diverse control tasks compared to A2C and PPO. Notably, both SA2C and SPPO perform well in settings with added noise to the reward. Additionally, SPPO shows consistent performance improvements across various hyperparameters (Table 4). We analyze why SPPO exhibits more robust improvements than SA2C in Section 5.4. Furthermore, SPPO shows superior performance to PPO in RLHF tasks, such as IMDB positive sentiment and TL;DR summarization. We demonstrate SPPO outperforming PPO on reward in both tasks, and SPPO's summarization is significantly better, as measured by win-rate against PPO, judged by GPT-4 Turbo (`gpt-4-turbo-2024-04-09`).

In summary, our key contributions are:

- We propose the symmetric RL loss for A2C and PPO, along with the gradient analysis that aligns with the gradient behavior of robust loss functions used in noisy classification tasks in Section 4.3.
- We conduct experiments across various environments and model scales, demonstrating performance improvements to validate the effectiveness of the symmetric RL loss for general control tasks and RLHF tasks in Section 5.
- We analyze how PPO can introduce additional confusion in advantage estimates, which justifies using symmetric RL loss (See Section 5.4). This shows that SPPO demonstrates consistent improvement across a range of hyperparameters.

2 RELATED WORK

We briefly introduce robust loss functions studied in the context of noise in supervised learning classification tasks. Ghosh et al. (2017) prove that, in the presence of a noisy dataset, the mean absolute error (MAE) has a slower learning speed compared to cross-entropy loss (CE), but the model learns more robustly. Zhang & Sabuncu (2018) propose a generalized cross entropy loss L_q, which becomes CE when $q \to 0$, and becomes MAE when $q \to 1$. By adjusting this parameter $0 \leq q \leq 1$, robust learning is achieved in noisy datasets. The symmetric cross entropy (SCE) (Wang et al., 2019) that we mainly refer to suggests a symmetric cross-entropy loss. This loss not only considers the flow of information from the true distribution to the model's predictions but also incorporates information flowing in the reverse direction. SCE works better than GCE in general, especially for data with high noise rates. Ma et al. (2020) introduce various loss functions and classify them into types: *Active Loss* and *Passive Loss* functions. They demonstrate that normalizing the loss can help improve robustness. They use a combination of one active loss and one passive loss like SCE. We define a loss function that considers reverse information to match the RL version and use it to improve the RL procedure.

In the RL literature, Wang et al. (2018) proposes using a confusion matrix to handle perturbed rewards, predicting surrogate rewards for robust policy updates. While this method appears effective for Atari games, later research (Chen et al., 2024) shows that it does not outperform corresponding baselines in continuous tasks. Additionally, introducing noise in RL has demonstrated performance benefits. For instance, Obando-Ceron et al. (2023) show that smaller batch sizes improve performance, and Schaul et al. (2022) present that policy churn aids exploration. These studies primarily conduct experiments on Atari games, which require navigating many novel states. However, whether noise is beneficial or not in continuous action spaces remains debatable (Mai et al., 2022; Byun & Perrault, 2024). Our work proposes a robust loss function designed to handle noise (confusion in advantage prediction) without judging whether the noise is beneficial.

Reinforcement Learning from Human Feedback (RLHF) Ouyang et al. (2022); Lee et al. (2023b) and Reinforcement Learning from AI Feedback (RLAIF) (Lee et al., 2023a; Bai et al., 2022) have contributed to the success of large language models (LLMs) by aligning them with user preferences. However, these methods require training a reward model and a value function. Each of these components has prediction errors, and finding appropriate hyperparameters for training requires significant effort. Direct Preference Optimization (DPO) (Rafailov et al., 2023) eliminates the cost associated with the reward model by rearranging PPO loss for ranking-based feedback (e.g., sample A is preferred over sample B). Ethayarajh et al. (2024) remove the requirement ranking-based feedback by modifying DPO loss further, allowing a model to be trained with bad or good labels. Additionally, Chakraborty et al. (2024) demonstrate that feedback from diverse people, each with different preferences, makes a single reward model difficult to reflect preferences correctly. Recent studies focus on sentence-level feedback (Lightman et al., 2023; Wang et al., 2024), but DPO and KTO cannot utilize sentence-level feedback. Therefore, we propose the reverse RL loss term, which can make PPO in existing RLHF methods more robust.

3 PRELIMINARIES

3.1 REINFORCEMENT LEARNING

Reinforcement Learning (RL) formulates a Markov decision process (MDP) (Puterman, 2014; Sutton & Barto, 2018b) defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma, \mu)$. At each timestep t, an action $a_t \in \mathcal{A}$ is sampled from an agent's policy $\pi_\theta(\cdot \mid s_t)$ for a given state $s_t \in \mathcal{S}$. For the taken action a_t, the reward function returns a reward $\mathcal{R}(s_t, a_t)$ where $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and the transition probability $\mathcal{P}(\cdot \mid s_t, a_t)$ determines the next state s_{t+1}. γ is the discount factor, and μ represents the initial state distribution for s_0. The RL objective is to find the optimal θ that maximizes the expected discounted sum of rewards:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \underset{\substack{s_0 \sim \mu \\ a_t \sim \pi_\theta(\cdot|s_t) \\ s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)}}{\mathbb{E}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right]. \tag{1}$$

3.2 A2C and PPO algorithms

The Advantage Actor-Critic (A2C) algorithm (Mnih et al., 2016) is an actor-critic method that combines value-based and policy-based approaches. A2C uses the advantage function A to reduce the variance in policy updates. The policy π_θ is updated by following the gradient of the objective function to maximize the sum of rewards as defined in 1:

$$\nabla_\theta J(\pi_\theta) = \sum_{t=0} \nabla_\theta \log \pi_\theta(a_t \mid s_t) A(s_t, a_t) \tag{2}$$

Proximal Policy Optimization (PPO) (Schulman et al., 2017) aims to update the policy within a trust region. This is achieved through a clipped loss function to ensure that the new policy does not deviate too much from the old policy. The PPO loss function can be written as:

$$L_{\text{ppo}}(\theta) = \mathbb{E}_t \left[\min\left(\mathrm{r_t}(\theta)\mathrm{A_t}, \mathrm{clip}(\mathrm{r_t}(\theta), 1-\epsilon, 1+\epsilon)\mathrm{A_t}\right)\right] \tag{3}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio, and ϵ is a small hyperparameter that controls the range of the clipping. The advantage function estimates how much better an action a is compared to the other actions for at a given state s. Both algorithms increase the probability of a for s if the corresponding advantage $A(a, s) > 0$ and decrease it if $A(a, s) < 0$. In the approach section, we introduce the connection between A2C and PPO with the cross-entropy loss for classification and define the symmetric RL loss.

3.3 Symmetric cross entropy

Symmetric Cross Entropy (SCE) (Wang et al., 2019) is designed for noisy classification datasets. Cross Entropy (CE) loss (Equation 4) performs effectively when the data is clean; however, it encounters challenges in the presence of noise. Given a true distribution q and a predicted distribution p, p is learned based on the information derived from q according to information theory. However, when q is noisy, p can only approximate the true distribution to a limited extent. To address this issue, Symmetric Cross Entropy (SCE) also consider incorporates information in the opposite direction through Reverse Cross Entropy (RCE) (Equation 5).

$$L_{\text{ce}} = -\sum_{k=1}^{K} q(k|\mathbf{x}) \log p(k|\mathbf{x}) \tag{4}$$

$$L_{\text{rce}} = -\sum_{k=1}^{K} p(k|\mathbf{x}) \log q(k|\mathbf{x}) \tag{5}$$

where $k \in \{1, \ldots, K\}$ is a class and $\mathbf{x}$ is an input. RCE loss has been proven to be robust to a certain amount of noise, but the learning speed is too slow. Therefore, SCE combines CE and RCE losses (Equation 6),

$$L_{\text{sce}} = \alpha L_{\text{ce}} + \beta L_{\text{rce}} \tag{6}$$

where α and β are constants determining the contribution of each part. SCE demonstrates performance improvement across various noisy ratios and types. As mentioned in the introduction section, the RL training process can lead to noisy advantage predictions, so we propose a symmetric RL loss in the next approach section.

4 Approach

This section introduces the reverse RL loss and proposes the symmetric RL loss for A2C (Mnih et al., 2016) and PPO (Schulman et al., 2017), an RL version of Symmetric Cross Entropy (SCE) (Wang et al., 2019). A2C and PPO training procedures basically increase or decrease the probability of an action depending on the advantage sign, but the advantage prediction involves noise due to several factors. A highly engineered reward function is required to eliminate errors, and the trained reward model has a prediction error in RLHF (Ouyang et al., 2022) and RLAIF (Lee et al., 2023a; Bai et al., 2022). Receiving feedback from multiple sources further complicates the training of the reward model (Chakraborty et al., 2024). Additionally, the value function also has model errors,

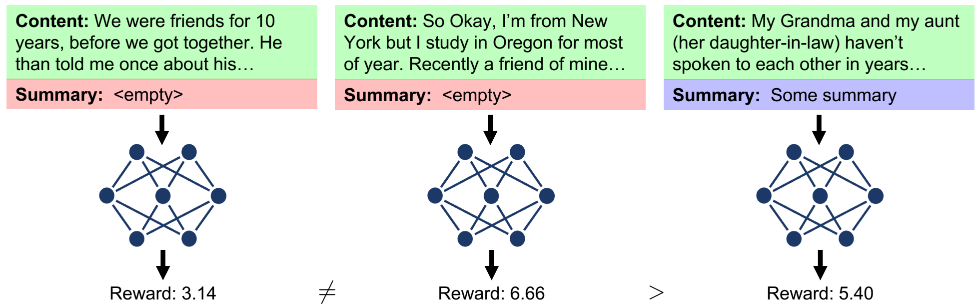

Figure 1: Example of reward prediction errors in a trained reward model for TL;DR summarization. The generated summary samples (left and middle) are both *empty*, yet they receive significantly different rewards. The middle sample is higher than some summarization text (right) and even scores higher (6.66) than the average reward score of SPPO (6.13). The full text for these samples can be found in Appendix 15.

and the sign of the advantage in advantage normalization depends on how the batch is composed. PPO increases sample efficiency compared to A2C, but the off-policy part can introduce confusion in advantage predictions (See Section 5.4). Similar to SCE, which is robust to noisy data, the symmetric RL loss contributes to robust learning in an RL environment that can introduce noisy predictions.

4.1 Reverse reinforcement learning loss

Given a true (target) distribution q and a predicted distribution p, if q is noisy, training p can be challenging and p cannot accurately reflect the true distribution. Reverse Cross Entropy (RCE) considers the reverse information from p. We propose that the reverse RL losses for A2C and PPO also incorporate reverse information to address noisy factors in the RL training procedure. The RCE loss (Equation 5) defines $\log 0 = Z$ where $Z < 0$ is some constant for $q(k|\mathbf{x}) = 0$. We also use this definition for the negative advantage and this also is useful to prove the robustness of the reverse RL losses. For all tasks conducted in this paper, we use $Z = -1$. Note that the constant terms Z and β in Equation 4 and 9 are multiplied together, so we control the impact of the reverse RL loss solely by adjusting β. For example, $(\beta = 1.0, Z = -1.0)$ and $(\beta = 10.0, Z = -0.1)$ yield the exact same results. Suppose there exist k actions and $a^{(i)}$ indicates i^{th} action. $\pi_\theta^{(i)} = \pi_\theta(a^{(i)}|s)$ for a state s. Let's denote the possible action probabilities set s as $\pi_\theta(s) = \{\pi_\theta^{(1)}, \pi_\theta^{(2)}, ..., \pi_\theta^{(k)}\}$. Note that we discretize the continuous action space for continuous action tasks (Tang & Agrawal, 2020). One thing we need to note is that when updating a policy, we use advantages instead of label sets in RL. *Advantages can have negative values* (negative labels) unlike ordinary labels. We only consider the sign of the advantage[1] because this advantage is the role of the label in supervised learning. For a sampled action probability $\pi_\theta^{(i)}$ and the corresponding advantage $A(s, a^{(i)}) = A^{(i)}$, the *sample-wise reverse A2C (RA2C) loss* is:

$$L_{\text{ra2c}}(\pi_\theta(s), A^{(i)}) = \begin{cases} \sum_{j \in [k] \setminus \{i\}} -\pi_\theta^{(j)} A^{(i)} Z, & \text{if } A^{(i)} > 0 \\ \sum_{j \in [k] \setminus \{i\}} \pi_\theta^{(j)} A^{(i)} Z, & \text{if } A^{(i)} < 0 \end{cases} \tag{7}$$

For a positive advantage A, the difference between A2C's loss $A \log \pi$ and CE loss $1 \log p$ is that A2C can be considered as CE multiplied by the advantage. In terms of gradients, A is a constant, so A2C reflects the information A times more strongly than the CE loss. Thus, we also reflect the reverse direction A times more strongly. Similarly, since PPO has $\pi_{\text{old}}^{(i)}$ term in the loss, the sample-wise reverse PPO (RPPO) loss just introduces the additional constant $\pi_{\text{old}}^{(i)}$ for a sampled action

[1] We do not consider when the advantages are zero because those are not considered when updating a policy.

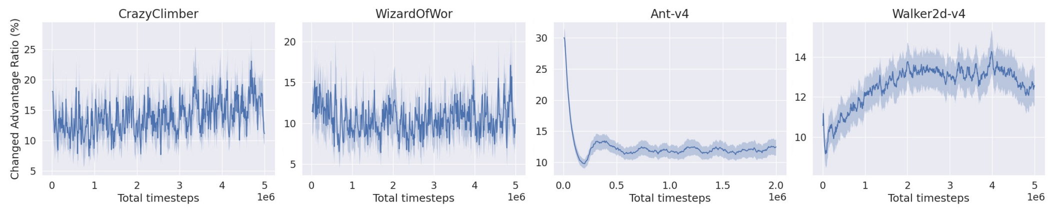

Figure 2: Change of advantage rate (%): The graphs show how often the advantage signs flip in various environments as training progresses. In Atari games, often over 5% of samples change signs, while in MuJoCo tasks, usually over 10% of samples change signs after the advantage normalization. We use 5 different random seeds for CrazyClimber and WizardOfWor, and 30 different random seeds for Ant-v4 and Walker2d-v4. The line is the mean of the change ratio across the seeds, and the shaded area represents standard errors.

probability $\pi_\theta^{(i)}$ to consider the same amount of reverse information:

$$L_{\text{rppo}}(\pi_\theta(s), A^{(i)}, \pi_{\text{old}}^{(i)}) = \begin{cases} \sum_{j\in[k]\setminus\{i\}} -\frac{\pi_\theta^{(j)} A^{(i)} Z}{\pi_{\text{old}}^{(i)}}, & \text{if } A^{(i)} > 0 \\ \sum_{j\in[k]\setminus\{i\}} \frac{\pi_\theta^{(j)} A^{(i)} Z}{\pi_{\text{old}}^{(i)}}, & \text{if } A^{(i)} < 0 \end{cases} \tag{8}$$

We define the symmetric RL loss, which is composed of the original RL loss (A2C or PPO) and the corresponding reverse RL loss, in Section 4.2. We then analyze why these reverse RL losses contribute to the RL learning procedure in Section 4.3.

4.2 Symmetric reinforcement learning loss

The *Symmetric Reinforcement Learning (SRL) loss* L_{srl} consists of two parts like SCE (Equation 6): the original actor loss L_{rl} (A2C or PPO) and the corresponding reverse RL loss L_{rev} (RA2C or RPPO). L_{srl} flexibly adjust the symmetric learning framework with two additional hyperparameters ($\alpha > 0$ and $\beta > 0$) as follows:

$$L_{\text{srl}} = \alpha L_{\text{rl}} + \beta L_{\text{rev}} \tag{9}$$

We name A2C and PPO using the symmetric RL loss as *Symmetric A2C (SA2C)* and *Symmetric PPO (SPPO)*, respectively. The meanings of α and β align with SCE, where α represents the degree of actively training a policy, and β serves as auxiliary support to stabilize the entire learning process. In the following section, we analyze the gradient of the two types of losses.

4.3 Gradient analysis

For an input $\mathbf{x}$ and the corresponding correct label k, the cross entropy (CE) loss gradient is $-\frac{1}{p_\theta(k|\mathbf{x})}\nabla_\theta p_\theta(k|\mathbf{x})$. Smaller p_θ values aggressively increase the magnitude of the gradient. CE loss rapidly increases uncertain predictions. If there is no noise, this method is correct, but it may lead to incorrect predictions on noisy datasets and excessive overfitting (Zhang & Sabuncu, 2018). A2C and PPO losses also have the same issue. For A2C, the gradient is simply multiplied by an advantage A, i.e., $-\frac{A(s,a)}{\pi_\theta(a|s)}\nabla_\theta \pi_\theta(a|s)$. In the case of PPO, the magnitude of the gradient tends to increase as the probability of an action decreases. Consider a sample that passes the clipping function: the difference between π_{old} and π is within the ϵ bound. As the denominator π_{old} gets smaller, the magnitude of the gradient increases.

Detailed analysis: The symmetric RL loss gradient analysis aligns with the analysis of SCE. For simplicity, we set α and β to 1 and examine the gradient direction for two types of A2C loss (RL and reverse RL) with respect to the action logits z. We use the notation defined in Section 4.1 and introduce the case when $A^{(i)} > 0$. For the full derivation including SPPO and $A^{(i)} < 0$, please refer to Appendix A. The sample-wise SA2C loss is:

$$L_{\text{sa2c}} = L_{\text{a2c}} + L_{\text{ra2c}} \tag{10}$$

Table 1: Mean final scores and standard errors (over the last 10 episodes) of PPO and SPPO on Atari games, without and with binary symmetric channel (BSC) noise with a crossover probability of 0.1 across 5 seeds. Full results can be found in Table 11.

	Without Noise		$\epsilon \sim$ **BSC**(0.1)	
	PPO	**SPPO**	PPO	**SPPO**
Alien	**1128 ± 105**	1081 ± 79	525 ± 26	**713 ± 26**
Centipede	2961 ± 379	**3694 ± 224**	4759 ± 257	**7525 ± 769**
CrazyClimber	86764 ± 3568	**103588 ± 2871**	71144 ± 11060	**99810 ± 2487**
Gravitar	371 ± 47	**442 ± 67**	269 ± 39	**332 ± 61**
Qbert	4352 ± 128	**4412 ± 282**	2827 ± 1927	**4020 ± 2415**
MsPacman	837 ± 62	**1204 ± 86**	704 ± 41	**1011 ± 52**
NameThisGame	**5665 ± 280**	5423 ± 63	2681 ± 143	**5187 ± 247**
UpNDown	58289 ± 21226	**126830 ± 27534**	8815 ± 1395	**73490 ± 33553**

The gradients for each part are:

$$\frac{\partial L_{\mathrm{a2c}}(\pi^{(i)}, A^{(i)})}{\partial z_y} = \begin{cases} A^{(i)}(\pi^{(i)} - 1), & \text{if } i = y \\ A^{(i)}\pi^{(y)}, & \text{if } i \neq y \end{cases} \tag{11}$$

$$\frac{\partial L_{\mathrm{ra2c}}(\pi^{(i)}, A^{(i)})}{\partial z_y} = \begin{cases} -A^{(i)}Z\pi^{(y)}(\pi^{(y)} - 1), & \text{if } i = y \text{ and } A^{(i)} > 0 \\ -A^{(i)}Z\pi^{(y)}\pi^{(i)}, & \text{if } i \neq y \text{ and } A^{(i)} > 0 \end{cases} \tag{12}$$

Thus, the SA2C loss gradient is:

$$\frac{\partial L_{\mathrm{sa2c}}}{\partial z_y} = \begin{cases} \underbrace{A^{(i)}(\pi^{(i)} - 1)}_{\nabla L_{\mathrm{a2c}} < 0} \underbrace{-A^{(i)}Z\pi^{(i)}(\pi^{(i)} - 1)}_{\nabla L_{\mathrm{ra2c}} < 0}, & \text{if } i = y \text{ and } A^{(i)} > 0 \\ \underbrace{A^{(i)}\pi^{(y)}}_{\nabla L_{\mathrm{a2c}} > 0} \underbrace{-A^{(i)}Z\pi^{(y)}\pi^{(i)}}_{\nabla L_{\mathrm{ra2c}} > 0}, & \text{if } i \neq y \text{ and } A^{(i)} > 0 \end{cases} \tag{13}$$

For both cases, the gradient directions of the RL (A2C) loss and the reverse RL (RA2C) loss are aligned. When $i = y$ and $A^{(i)} > 0$, the gradient of the RA2C loss is $-A^{(i)}Z\pi^{(y)}(\pi^{(y)} - 1)$, reaching its maximum magnitude at $\pi^{(y)} = 0.5$ as a parabolic function. This means that the accelerator helps the probability $\pi^{(i)}$ increase most rapidly when the action to take is ambiguous. When $i \neq y$ and $A^{(i)} > 0$, the probability of actions other than $a^{(i)}$ is reduced, and this reduction is influenced by the confidence of both $\pi^{(i)}$ and $\pi^{(y)}$. Specifically, the gradient of the RA2C loss is $-A^{(i)}Z\pi^{(y)}\pi^{(i)}$. When both $\pi^{(i)}$ and $\pi^{(y)}$ are 0.5, representing the most ambiguous predictions, the accelerator aids the A2C loss in reducing $\pi^{(y)}$ most effectively. Thus, the RA2C loss helps deviate from ambiguous predictions as an accelerator. SPPO's loss gradients are also aligned like SA2C and follow the same mechanism (See Appendix B.2).

5 Experiments

To validate the effectiveness of our algorithm, we conduct experiments on various tasks and models of different scales. First, we experiment on Atari games (Mnih et al., 2013) featuring discrete action spaces (Section 5.1), as well as MuJoCo benchmark tasks (Todorov et al., 2012) and Box2D tasks (Catto, 2011) (Section 5.2) with continuous action spaces using Stable-Baselines3 (Raffin et al., 2021). In these control tasks, we also create a variant of each that introduces reward noise, hypothesizing that it will create more confusion in advantage prediction. SPPO performs better than SA2C for various reverse RL loss hyperparameters β. We also evaluate our method on IMDB and TL;DR datasets using TRIL Chang et al. (2023) to determine whether our approach is practical for LLM tasks. We primarily present the experimental results for PPO in the main paper. In the latter part of this section, we analyze why our method works better with PPO than A2C (Section 5.4), conduct hyperparameter sensitivity tests, and examine the training cost (Section 5.5).

Table 2: Mean final scores and standard errors (over the last 10 episodes) of PPO and SPPO on Atari games, without and with binary symmetric channel (BSC) noise with a crossover probability of 0.1 across 5 seeds. To leverage the reverse RL loss, we discretize the continuous action space. DPPO is added as another baseline ($\alpha = 1.0, \beta = 0.0$), and DSPPO is our proposed method. Full results can be found in Table 12 and 13.

$\epsilon \sim \mathcal{N}(0, 0.05^2)$	**Ant**	**Hopper**	**HalfCheetah**	**HumanoidStandup**
PPO	601 ± 47	1936 ± 147	2068 ± 208	80945 ± 2130
DPPO	1897 ± 86	2153 ± 106	2722 ± 188	$\mathbf{146038 \pm 1841}$
DSPPO	$\mathbf{2095 \pm 102}$	$\mathbf{2333 \pm 109}$	$\mathbf{3118 \pm 195}$	145974 ± 2520
	Walker2d	**Swimmer**	**BipedalWalker**	**LunarLanderContinuous**
PPO	1270 ± 107	44 ± 3.0	158 ± 15.2	181 ± 13.8
DPPO	3419 ± 100	57 ± 3.6	$\mathbf{274 \pm 7.1}$	281 ± 5.7
DSPPO	$\mathbf{3523 \pm 129}$	$\mathbf{72 \pm 5.1}$	267 ± 8.8	$\mathbf{294 \pm 3.3}$

5.1 Discrete action space tasks

We first conduct experiments on Atari games (Mnih et al., 2016) that the action spaces are discrete to evaluate SPPO and SA2C. We primarily select 22 games based on the reported score for A2C in Schulman et al. (2017), focusing on games where the A2C scores are not close to 0, as this allows us to demonstrate meaningful score changes.

The reward functions for Atari games only output 0 or 1 and are well-defined. To introduce some reward noise, we flip the reward from 0 to 1 or from 1 to 0 with a probability of 10%. We denote this noise setting as a Binary Symmetric Channel (BSC). This setting is analogous to a potential problem in ranking-based feedback (Ouyang et al., 2022) from humans or AI, where evaluators may have different preferences, resulting in reversed scores. We observe that SA2C shows marginal improvements (Table 8), with a narrow range of effective hyperparameter β values. In contrast, SPPO performs well in both noise-free and noisy environments (See Section 5.4 for discussion). Table 1 presents partial results, while the complete results for SPPO, including training curves (Figure C.3), can be found in Table 11. SPPO achieves 16 out of 22 wins in noise-free settings and 19 out of 22 wins in noisy settings.

5.2 Continuous action space tasks

Next, we perform experiments on MuJoCo benchmark (Todorov et al., 2012) and Box2D (Catto, 2011) continuous action space environments. To utilize the reverse RL loss, we need other action probabilities for a sampled action probability. However, conventional RL uses a multivariate Gaussian distribution as a policy, so it cannot provide the other action probabilities. Thus, we discretize the continuous action space (Tang & Agrawal, 2020), naming these methods DA2C and DPPO, and add them as additional baseline comparisons.

Note that discretizing the continuous action space generally works better than the original RL methods like A2C and PPO for these tasks if the continuous action space is discretized with a sufficient number of bins. This discretized distribution can represent more complex distributions than a diagonal Gaussian distribution (where the covariance is diagonal). We apply the reverse RL loss to both DA2C and DSPPO.

Since the reward functions in these environments are highly engineered, we perturb the reward function with Gaussian noise with a mean of 0 and a standard deviation of 0.05. Table 2 shows partial results for SPPO under noise settings. The full experiment results are in Table 12 and 13. Similar to the Atari game results, SA2C without noise shows tied performance in the noiseless setting, and improvements when the reward noise is introduced. SPPO consistently shows robust performance gains across a wide range of β values for both settings.

Table 3: RM Score indicates the reward model score, Perplexity measures the uncertainty of the model, and Win Rate is judged by GPT-4 Turbo by comparing the generated output and reference text. We use 4 different random seeds for each task.

	IMDB Sentiment		TL;DR Summarization		
	RM Score (↑)	Perplexity (↓)	RM Score (↑)	Perplexity (↓)	Win Rate (↑)
SFT	0.54 ± 0.00	33.02 ± 0.09	5.83 ± 0.02	18.35 ± 0.02	42.00 ± 2.58
PPO	0.89 ± 0.02	41.09 ± 0.43	5.94 ± 0.08	19.08 ± 0.17	43.25 ± 3.82
SPPO	$\mathbf{0.92 \pm 0.01}$	40.60 ± 0.44	$\mathbf{6.13 \pm 0.02}$	19.27 ± 0.21	$\mathbf{52.50 \pm 2.40}$

5.3 RLHF TASKS

The final tasks are RLHF tasks to determine if our method is applicable to large language models. The first task is IMDB positive sentiment. The objective of the IMDB task is to generate positive sentiment continuations for movie reviews (Maas et al., 2011). The sentiment classifier (Sanh et al., 2019) is used as a reward model to evaluate how positive a provided text is. The base policy is GPT-2 (Radford et al., 2019), which we fine-tune using PPO or SPPO. We evaluate this model based on the reward score and perplexity. SPPO shows improvement in both reward score and perplexity compared to PPO.

The second RLHF task is TL;DR summarization (Völske et al., 2017). The objective is to summarize a post from Reddit. The reward model is a fine-tuned GPT-J (Wang & Komatsuzaki, 2021) with LoRA adapters (Hu et al., 2021) by Chang et al. (2023). The training dataset for this reward model is the filtered dataset with additional human preference data used in Stiennon et al. (2020). The base policy model is an open-source GPT-J model (`CarperAI/openai_summarize_tldr_sft`) with added LoRA adapters. Note that the open-source GPT-J model is often outputs empty summarizations for most evaluation data. Therefore, we report results after 10 epochs of RL updates as an alternative to SFT, as it begins to consistently summarize posts. We evaluate SPPO based on the reward score, perplexity, and win rate. This win rate is judged by GPT-4 Turbo (OpenAI, 2024) (`gpt-4-turbo-2024-04-09`) by comparing the generated output and reference text. Even though the perplexity of SPPO is slightly higher than that of PPO, there is an improvement in the reward score and a significantly increased win rate.

In the introduction section, we mention that RLHF or RLAIF have additional errors due to a trained reward model. We check whether the trained reward model used in TL;DR has reward prediction errors. Figure 4.1 shows a dramatic example: the generated summary sample (left) and the middle sample were both *empty*, but the two rewards show a huge gap. The middle sample even scores (6.66) better than those learned with an SPPO score (6.13). Wrong summaries, like *empty*, can score higher than a summarized text (right). These cases are observed very often. This makes the RL training procedure more noisy and means that the sign of advantage changes depending on how the batch is composed. The full text for these samples can be found in Appendix D.

5.4 WHY SPPO WORKS BETTER THAN SA2C

The motivation for using the reverse RL loss is to address the issue of ambiguity in advantage predictions (Section 4.3). We hypothesize that the PPO advantage prediction (sign) is less consistent than in A2C during policy updates, but this does not mean that PPO is worse than A2C. There are two main reasons why consistency is not maintained. First, PPO has improved sample efficiency compared to A2C, but after the first epoch, subsequent updates become off-policy, affecting advantage estimates. Second, PPO often uses advantage normalization to restrict large advantage values from being involved with policy updates to stabilize the learning process. In addition, PPO often uses smaller mini-batch sizes (e.g., 64), whereas A2C uses the entire dataset for policy updates. Many popular RL code baselines, such as Stable Baselines3 (Raffin et al., 2021), RL4LMs (Ramamurthy et al., 2023), TRL (von Werra et al., 2020), and TRLX (Havrilla et al., 2023) use PPO advantage normalization by default, whereas A2C does not. Our experiments on the usefulness of advantage normalization also show that the performance increase in IMDB is greater than the performance decrease in TL;DR (Appendix 14).

Table 4: Percentage improvement of SPPO over PPO. The percentage improvements are computed across 22 Atari games. We simply fix $\alpha = 0.5$ to reduce the total loss magnitude and vary β to control the impact of the reverse RL loss. We exclude very large improvements (e.g., 2000%) from calculating the average. This large improvements result from PPO's significant learning failures.

$\alpha = 0.5$ is fixed	$\beta = 0.5$	$\beta = 1.0$	$\beta = 5.0$	$\beta = 10.0$	$\beta = 25.0$
SPPO under 0% noise	7.83%	10.15%	24.98%	21.52%	18.92%
SPPO under 10% noise	1.74%	21.89%	148.46%	166.73%	136.50%

We examine the ratio of advantage sign changes before and after normalization for PPO in Atari games and MuJoCo tasks (Figure 2). This ratio varies across different environments. The advantage sign changes usually exceed 5% for Atari games and 10% for MuJoCo and Box2D environments. These changes introduce the confusion, which makes the reverse RL loss more effective for PPO. This observation aligns with our motivation for using symmetric RL loss to handle noisy data, similar to how it is addressed in noisy classification tasks in supervised learning.

Additionally, since A2C uses the entire dataset (rather than using advantage normalization with small batches) for the policy updates, it introduces less confusion in advantage prediction. As a result, SA2C demonstrates performance comparable to A2C in settings without reward noise (Table 8 and 9), and improvements in settings with reward noise (Table 8 and 10), where advantage estimation is more likely to be confused.

5.5 Hyperparameters and Training Cost

Although the symmetric RL loss introduces three additional hyperparameters (Equation 4.2): α, β, and Z, we simply fix $\alpha = 0.5$ in all experiments to reduce the overall magnitude of the symmetric loss. Additionally, since β and Z are constants that are multiplied together, we can fix one and adjust the other. For example, $(\beta = 1.0, Z = -1.0)$ and $(\beta = 10.0, Z = -0.1)$ yield the same results. In our experiments, we fix $Z = -1$ and adjust β to determine the influence of the reverse RL loss.

We test the sensitivity of β for SPPO on Atari games with and without noise in the rewards. Table 5 presents the percentage improvements compared to PPO. We exclude excessively large improvements (e.g., 2000%) to avoid skewing the average. These significant improvements typically result from PPO's training failure, while SPPO remains stable (Gopher and WizardOfWor in Figure C.3). Fixing $\alpha = 0.5$ and $Z = -1$, we vary β and observe consistent improvements, demonstrating SPPO's robustness across hyperparameters. Also, we use the default values of Stable Baselines3 (Raffin et al., 2021) for the other RL hyperparameters; more details can be found in Appendix C.1.

The symmetric RL loss introduces the reverse RL loss term, which is essentially another form of cross-entropy that does not significantly increase training time. In practice, there is no increase in training time for the continuous tasks discussed in Section 5.2 and the LLM tasks in Section 5.3, and a 10–20% increase for the Atari games in Section 5.1.

6 Conclusion and Future Work

We present Symmetric RL loss, inspired by Symmetric Cross Entropy (SCE) (Wang et al., 2019) from supervised learning, to enhance the robustness of RL. SCE leverages reverse information to handle noisy data, which we adapt to RL algorithms like A2C and PPO, resulting in SA2C and SPPO. We test SA2C and SPPO on various discrete and continuous action space tasks and further evaluate SPPO on RLHF tasks, including IMDB positive sentiment and TL;DR summarization. Our results show that SPPO consistently outperforms PPO. We aregue this is mainly due to PPO's off-policy parts and advantage normalization with small batch sizes, which lead to advantage sign changes (confusion). The Symmetric RL loss for SPPO alleviates this training difficulty.

While we only propose the Symmetric RL loss specifically for A2C and PPO, exploring its integration into other RL algorithms, such as DQN or SAC, is an intriguing future direction. Additionally, developing more diverse reverse RL loss functions, like the Normalized Loss Functions (Ma et al., 2020) proposed after SCE, is also an interesting direction for future work.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms, 2024.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional ai: Harmlessness from ai feedback, 2022.

Ju-Seung Byun and Andrew Perrault. Normality-guided distributional reinforcement learning for continuous control, 2024. URL `https://arxiv.org/abs/2208.13125`.

Ju-Seung Byun, Jiyun Chun, Jihyung Kil, and Andrew Perrault. Ares: Alternating reinforcement learning and supervised fine-tuning for enhanced multi-modal chain-of-thought reasoning through diverse ai feedback, 2024. URL `https://arxiv.org/abs/2407.00087`.

Erin Catto. Box2d, a 2d physics engine for games, 2011. URL `http://box2d.org`.

Souradip Chakraborty, Jiahao Qiu, Hui Yuan, Alec Koppel, Furong Huang, Dinesh Manocha, Amrit Singh Bedi, and Mengdi Wang. Maxmin-rlhf: Towards equitable alignment of large language models with diverse human preferences, 2024.

Jonathan D Chang, Kiante Brantley, Rajkumar Ramamurthy, Dipendra Misra, and Wen Sun. Tril: Transformers reinforcement and imitation learning library. `https://github.com/Cornell-RL/tril`, 2023.

Xi Chen, Zhihui Zhu, and Andrew Perrault. The distributional reward critic architecture for perturbed-reward reinforcement learning, 2024. URL `https://arxiv.org/abs/2401.05710`.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models, 2022.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model, 2023.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Kto: Model alignment as prospect theoretic optimization, 2024.

Aritra Ghosh, Himanshu Kumar, and P. S. Sastry. Robust loss functions under label noise for deep neural networks, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

Alexander Havrilla, Maksym Zhuravinskyi, Duy Phung, Aman Tiwari, Jonathan Tow, Stella Biderman, Quentin Anthony, and Louis Castricato. trlX: A framework for large scale reinforcement learning from human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8578–8595, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.530. URL https://aclanthology.org/2023.emnlp-main.530.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685, 2021. URL https://arxiv.org/abs/2106.09685.

Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. Clinicalbert: Modeling clinical notes and predicting hospital readmission, 2020.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents, 2022.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. Rlaif: Scaling reinforcement learning from human feedback with ai feedback, 2023a.

Kimin Lee, Hao Liu, Moonkyung Ryu, Olivia Watkins, Yuqing Du, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, and Shixiang Shane Gu. Aligning text-to-image models using human feedback, 2023b.

Peter Lee, Sebastien Bubeck, and Joseph Petro. Benefits, limits, and risks of gpt-4 as an ai chatbot for medicine. *New England Journal of Medicine*, 388(13):1233–1239, 2023c. doi: 10.1056/NEJMsr2214184. URL https://www.nejm.org/doi/full/10.1056/NEJMsr2214184.

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, 2023.

Xingjun Ma, Hanxun Huang, Yisen Wang, Simone Romano, Sarah M. Erfani, and James Bailey. Normalized loss functions for deep learning with noisy labels. *CoRR*, abs/2006.13554, 2020. URL https://arxiv.org/abs/2006.13554.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea (eds.), *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL https://aclanthology.org/P11-1015.

Vincent Mai, Kaustubh Mani, and Liam Paull. Sample efficient deep reinforcement learning via uncertainty estimation, 2022. URL https://arxiv.org/abs/2201.01666.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL http://arxiv.org/abs/1312.5602.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783, 2016. URL http://arxiv.org/abs/1602.01783.

Johan Obando-Ceron, Marc G. Bellemare, and Pablo Samuel Castro. Small batch deep reinforcement learning, 2023. URL https://arxiv.org/abs/2310.03882.

OpenAI. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

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, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2023.

Antonin Raffin. Rl baselines3 zoo. `https://github.com/DLR-RM/rl-baselines3-zoo`, 2020.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL `http://jmlr.org/papers/v22/20-1364.html`.

Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. Is reinforcement learning (not) for natural language processing: Benchmarks, baselines, and building blocks for natural language policy optimization, 2023.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019. URL `http://arxiv.org/abs/1910.01108`.

Tom Schaul, André Barreto, John Quan, and Georg Ostrovski. The phenomenon of policy churn, 2022. URL `https://arxiv.org/abs/2206.00730`.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. 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. 1889–1897, Lille, France, 07–09 Jul 2015. PMLR. URL `https://proceedings.mlr.press/v37/schulman15.html`.

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

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2018.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F. Christiano. Learning to summarize from human feedback. *CoRR*, abs/2009.01325, 2020. URL `https://arxiv.org/abs/2009.01325`.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 2018a. ISBN 978-0262039246.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018b.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pp. 1057–1063, 2000.

Yunhao Tang and Shipra Agrawal. Discretizing continuous action space for on-policy optimization, 2020.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning, 2015.

Michael Völske, Martin Potthast, Shahbaz Syed, and Benno Stein. TL;DR: Mining Reddit to learn automatic summarization. In Lu Wang, Jackie Chi Kit Cheung, Giuseppe Carenini, and Fei Liu (eds.), *Proceedings of the Workshop on New Frontiers in Summarization*, pp. 59–63, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4508. URL https://aclanthology.org/W17-4508.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, and Shengyi Huang. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl, 2020.

Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

Jingkang Wang, Yang Liu, and Bo Li. Reinforcement learning with perturbed rewards. *CoRR*, abs/1810.01032, 2018. URL http://arxiv.org/abs/1810.01032.

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

Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jinfeng Yi, and James Bailey. Symmetric cross entropy for robust learning with noisy labels. *CoRR*, abs/1908.06112, 2019. URL http://arxiv.org/abs/1908.06112.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Zhilu Zhang and Mert R. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *CoRR*, abs/1805.07836, 2018. URL http://arxiv.org/abs/1805.07836.

A GRADIENT OF RL LOSS AND REVERSE RL LOSS

Suppose there exist k actions, and $a^{(i)}$ indicates the i^{th} action. Let $\pi_\theta^{(i)} = \pi_\theta(a^{(i)}|s)$ denote the policy for a state s. The set $\pi_\theta(s) = \{\pi_\theta^{(1)}, \pi_\theta^{(2)}, \ldots, \pi_\theta^{(k)}\}$ represents the possible action probabilities set for s. $A^{(i)}$ indicates the corresponding advantage of the sampled action $a^{(i)}$ for s. $Z < 0$ is a constant used in the reverse RL loss to handle the computational issue where $\log 0 = -\infty$. For simplicity of notation, we drop θ, s, and a from the policy π. Note that $A^{(i)}$ and Z are not involved with the gradient as they are constants with respect to θ.

A.1 A2C LOSS

The derivation of the A2C loss L_{a2c} with respect to logits z is presented as follows:

For $i = y$,

$$
\begin{aligned}
\frac{\partial \pi^{(i)}}{\partial z_y} &= \frac{\partial}{\partial z_y} \frac{e^{z_i}}{\sum_{w=1}^{k} e^{z_w}} \\
&= \frac{e^{z_i} \sum_{w=1}^{k} e^{z_w} - e^{z_i} e^{z_i}}{(\sum_{w=1}^{k} e^{z_w})^2} \qquad (14) \\
&= \pi^{(i)}(1 - \pi^{(i)})
\end{aligned}
$$

For $i \neq y$,

$$\frac{\partial \pi^{(i)}}{\partial z_y} = \frac{\partial}{\partial z_y} \frac{e^{z_i}}{\sum_{w=1}^{k} e^{z_w}} = -\frac{e^{z_i} e^{z_y}}{(\sum_{w=1}^{k} e^{z_w})^2} = -\pi^{(i)} \pi^{(y)} \tag{15}$$

The sample-wise A2C loss is:

$$L_{\mathrm{a2c}}(\pi^{(i)}, A^{(i)}) = -A^{(i)} \log \pi^{(i)} \tag{16}$$

For $i = y$,

$$\begin{aligned} \frac{\partial L_{\mathrm{a2c}}}{\partial z_y} &= \frac{\partial}{\partial z_y} - A^{(i)} \log \pi^{(i)} \\ &= -A^{(i)} \frac{\partial}{\partial z_y} \log \pi^{(i)} \\ &= -\frac{A^{(i)}}{\pi^{(i)}} \frac{\partial \pi^{(i)}}{\partial z_y} \\ &= A^{(i)}(\pi^{(i)} - 1) \text{ by (14)} \end{aligned} \tag{17}$$

For $i \neq y$,

$$\begin{aligned} \frac{\partial L_{\mathrm{a2c}}}{\partial z_y} &= \frac{\partial}{\partial z_y} - A^{(i)} \log \pi^{(i)} \\ &= -A^{(i)} \frac{\partial}{\partial z_y} \log \pi^{(i)} \\ &= -\frac{A^{(i)}}{\pi^{(i)}} \frac{\partial \pi^{(i)}}{\partial z_y} \\ &= A^{(i)} \pi^{(y)} \text{ by (15)} \end{aligned} \tag{18}$$

In summary, we have the following form for $L_{\mathrm{a2c}}(\pi^{(i)}, A^{(i)})$:

$$\frac{\partial L_{\mathrm{a2c}}(\pi^{(i)}, A^{(i)})}{\partial z_y} = \begin{cases} A^{(i)}(\pi^{(i)} - 1), & \text{if } i = y \\ A^{(i)} \pi^{(y)}, & \text{if } i \neq y \end{cases} \tag{19}$$

A.2 REVERSE A2C LOSS

The derivation of the reverse A2C loss L_{ra2c} with respect to logits z is presented as follows:

$$L_{\mathrm{ra2c}}(\pi^{(i)}, A^{(i)}) = \begin{cases} \sum_{j \in [k] \setminus \{i\}} -\pi^{(j)} A^{(i)} Z, & \text{if } A^{(i)} > 0 \\ \sum_{j \in [k] \setminus \{i\}} \pi^{(j)} A^{(i)} Z, & \text{if } A^{(i)} < 0 \end{cases} \tag{20}$$

For $i = y$ and $A^{(i)} > 0$,

$$\begin{aligned} \frac{\partial L_{\mathrm{ra2c}}}{\partial z_y} &= \frac{\partial}{\partial z_y} \sum_{j \in [k] \setminus \{i\}} -\pi^{(j)} A^{(i)} Z \\ &= -A^{(i)} Z \sum_{j \in [k] \setminus \{i\}} \frac{\partial \pi^{(j)}}{\partial z_y} \\ &= -A^{(i)} Z \sum_{j \in [k] \setminus \{i\}} -\pi^{(j)} \pi^{(y)} \text{ by (15)} \\ &= A^{(i)} Z \pi^{(y)} (1 - \pi^{(i)}) \\ &= -A^{(i)} Z \pi^{(y)} (\pi^{(y)} - 1) \end{aligned} \tag{21}$$

For $i \neq y$ and $A^{(i)} > 0$,

$$\begin{aligned}
\frac{\partial L_{\text{ra2c}}}{\partial z_y} &= \frac{\partial}{\partial z_y} \sum_{j \in [k] \setminus \{i\}} -\pi^{(j)} A^{(i)} Z \\
&= -A^{(i)} Z \sum_{j \in [k] \setminus \{i\}} \frac{\partial \pi^{(j)}}{\partial z_y} \\
&= -A^{(i)} Z \left(\sum_{j \in [k]} \frac{\partial \pi^{(j)}}{\partial z_y} - \frac{\partial \pi^{(i)}}{\partial z_y} \right) \\
&= A^{(i)} Z \left(\sum_{j \in [k]} -\pi^{(j)} \pi^{(y)} + \pi^{(y)} \pi^{(y)} + \pi^{(y)} (1 - \pi^{(y)}) - \pi^{(i)} \pi^{(y)} \right) \quad \text{by (14) and (15)} \\
&= -A^{(i)} Z \pi^{(y)} \pi^{(i)}
\end{aligned} \tag{22}$$

For $i = y$ and $A^{(i)} < 0$, the only difference from Equation 21 is the negative sign, thus:

$$\frac{\partial L_{\text{ra2c}}}{\partial z_y} = A^{(i)} Z \pi^{(y)} (\pi^{(y)} - 1) \quad \text{by (21)} \tag{23}$$

For $i \neq y$ and $A^{(i)} < 0$, the only difference from Equation 22 is the negative sign, thus:

$$\frac{\partial L_{\text{ra2c}}}{\partial z_y} = A^{(i)} Z \pi^{(y)} \pi^{(i)} \quad \text{by (22)} \tag{24}$$

In summary, we have the following form for $L_{\text{a2c}}(\pi^{(i)}, A^{(i)})$:

$$\frac{\partial L_{\text{ra2c}}(\pi^{(i)}, A^{(i)})}{\partial z_y} = \begin{cases} -A^{(i)} Z \pi^{(y)} (\pi^{(y)} - 1), & \text{if } i = y \text{ and } A^{(i)} > 0 \\ -A^{(i)} Z \pi^{(y)} \pi^{(i)}, & \text{if } i \neq y \text{ and } A^{(i)} > 0 \\ A^{(i)} Z \pi^{(y)} (\pi^{(y)} - 1), & \text{if } i = y \text{ and } A^{(i)} < 0 \\ A^{(i)} Z \pi^{(y)} \pi^{(i)}, & \text{if } i \neq y \text{ and } A^{(i)} < 0 \end{cases} \tag{25}$$

A.3 PPO LOSS

The derivation of the PPO loss L_{ppo} with respect to the logits z is presented as follows. The PPO loss includes a clipping function and a minimum operation. When these conditions are not satisfied, there is no gradient.

The sample-wise PPO loss is:

$$L_{\text{ppo}}(\pi^{(i)}, A^{(i)}, \pi_{\text{old}}^{(i)}) = -\frac{\pi^{(i)}}{\pi_{\text{old}}^{(i)}} A^{(i)} \tag{26}$$

For $i = y$,

$$\begin{aligned}
\frac{\partial L_{\text{ppo}}}{\partial z_y} &= \frac{\partial}{\partial z_y} - \frac{\pi^{(i)}}{\pi_{\text{old}}^{(i)}} A^{(i)} \\
&= -\frac{A^{(i)}}{\pi_{\text{old}}^{(i)}} \frac{\partial \pi^{(i)}}{\partial z_y} \\
&= \frac{A^{(i)} \pi^{(i)} (\pi^{(i)} - 1)}{\pi_{\text{old}}^{(i)}} \quad \text{by (14)}
\end{aligned} \tag{27}$$

For $i \neq y$,

$$\begin{aligned}\frac{\partial L_{\text{ppo}}}{\partial z_y} &= \frac{\partial}{\partial z_y} - \frac{\pi^{(i)}}{\pi_{\text{old}}^{(i)}} A^{(i)} \\ &= \frac{A^{(i)}}{\pi_{\text{old}}^{(i)}} \frac{\partial \pi^{(i)}}{\partial z_y} \\ &= \frac{A^{(i)} \pi^{(i)} \pi^{(y)}}{\pi_{\text{old}}^{(i)}} \quad \text{by (15)}\end{aligned} \tag{28}$$

A.4 Reverse PPO loss

The derivation of the reverse PPO loss L_{rppo} with respect to logits z is presented as follows. As with PPO, the reverse PPO loss only considers samples that pass the clipping function and the minimum operation.

From Section A.2, we have the following form for $L_{\text{rppo}}(\pi^{(i)}, A^{(i)}, \pi_{\text{old}}^{(i)})$:

$$\frac{\partial L_{\text{rppo}}(\pi^{(i)}, A^{(i)}, \pi_{\text{old}}^{(i)})}{\partial z_y} = \begin{cases} -\frac{A^{(i)} Z \pi^{(y)} (\pi^{(y)} - 1)}{\pi_{\text{old}}^{(i)}}, & \text{if } i = y \text{ and } A^{(i)} > 0 \\ -\frac{A^{(i)} Z \pi^{(y)} \pi^{(i)}}{\pi_{\text{old}}^{(i)}}, & \text{if } i \neq y \text{ and } A^{(i)} > 0 \\ \frac{A^{(i)} Z \pi^{(y)} (\pi^{(y)} - 1)}{\pi_{\text{old}}^{(i)}}, & \text{if } i = y \text{ and } A^{(i)} < 0 \\ \frac{A^{(i)} Z \pi^{(y)} \pi^{(i)}}{\pi_{\text{old}}^{(i)}}, & \text{if } i \neq y \text{ and } A^{(i)} < 0 \end{cases} \tag{29}$$

B Gradient analysis of RL loss and reverse RL loss

B.1 Symmetric A2C gradient analysis

The gradient analysis of the symmetric RL loss follows the SCE analysis. We adopt their analysis and extend it to cover the RL loss analysis. We set α and β to 1 for simplicity and evaluate the gradient direction of both RL and reverse RL losses with respect to the logits z. We show that the gradient directions for both types are the same and that the reverse RL loss helps deviate ambiguous predictions where the probability is around 0.5. We first show how the symmetric A2C (SA2C) loss behaves. Note that $Z < 0$ is a constant used in the reverse RL loss to handle $\log 0 = -\infty$.

$$L_{\text{sa2c}} = L_{\text{a2c}} + L_{\text{ra2c}} \tag{30}$$

For $i = y$ and $A^{(i)} > 0$,

$$\begin{aligned}\frac{\partial L_{\text{sa2c}}}{\partial z_y} &= \frac{\partial L_{\text{a2c}}}{\partial z_y} + \frac{\partial L_{\text{ra2c}}}{\partial z_y} \\ &= \underbrace{A^{(i)}(\pi^{(i)} - 1)}_{\nabla L_{\text{a2c}} < 0} \underbrace{- A^{(i)} Z \pi^{(i)} (\pi^{(i)} - 1)}_{\nabla L_{\text{ra2c}} < 0} \quad \text{by (17) and (21)}\end{aligned} \tag{31}$$

For $i \neq y$ and $A^{(i)} > 0$,

$$\begin{aligned}\frac{\partial L_{\text{sa2c}}}{\partial z_y} &= \frac{\partial L_{\text{a2c}}}{\partial z_y} + \frac{\partial L_{\text{ra2c}}}{\partial z_y} \\ &= \underbrace{A^{(i)} \pi^{(y)}}_{\nabla L_{\text{a2c}} > 0} \underbrace{- A^{(i)} Z \pi^{(y)} \pi^{(i)}}_{\nabla L_{\text{ra2c}} > 0} \quad \text{by (18) and (22)}\end{aligned} \tag{32}$$

For $i = y$ and $A^{(i)} < 0$,

$$\begin{aligned}\frac{\partial L_{\text{sa2c}}}{\partial z_y} &= \frac{\partial L_{\text{a2c}}}{\partial z_y} + \frac{\partial L_{\text{ra2c}}}{\partial z_y} \\ &= \underbrace{A^{(i)}(\pi^{(i)} - 1)}_{\nabla L_{\text{a2c}} > 0} \underbrace{-A^{(i)} Z \pi^{(y)}(\pi^{(y)} - 1)}_{\nabla L_{\text{ra2c}} > 0} \quad \text{by (17) and (21)}\end{aligned} \tag{33}$$

For $i \neq y$ and $A^{(i)} < 0$,

$$\begin{aligned}\frac{\partial L_{\text{sa2c}}}{\partial z_y} &= \frac{\partial L_{\text{a2c}}}{\partial z_y} + \frac{\partial L_{\text{ra2c}}}{\partial z_y} \\ &= \underbrace{A^{(i)}\pi^{(y)}}_{\nabla L_{\text{a2c}} < 0} \underbrace{-A^{(i)} Z \pi^{(y)}\pi^{(i)}}_{\nabla L_{\text{ra2c}} < 0} \quad \text{by (18) and (22)}\end{aligned} \tag{34}$$

For the above cases, the gradient directions of the RL (A2C) loss and the reverse RL (RA2C) loss are the same as SCE gradients. Essentially, the RA2C loss acts as an accelerator. In the case of $i = y$ and $A^{(i)} > 0$, the gradient of the RA2C loss is $-A^{(i)} Z \pi^{(y)}(\pi^{(y)} - 1)$, with the largest gradient magnitude at $\pi^{(y)} = 0.5$ as a parabolic function. In other words, the accelerator helps the probability $\pi^{(i)}$ increase most quickly when it is ambiguous which action to take. In the case of $i \neq y$ and $A^{(i)} > 0$, the probability of other actions except $a^{(i)}$ is reduced, and this reduction is influenced by the confidence of both $\pi^{(i)}$ and $\pi^{(y)}$. Specifically, the gradient of the RA2C loss is $-A^{(i)} Z \pi^{(y)}\pi^{(i)}$. When both $\pi^{(i)}$ and $\pi^{(y)}$ are 0.5, indicating the most ambiguous predictions, the accelerator helps the A2C loss reduce $\pi^{(y)}$ most aggressively.

When $A^{(i)} < 0$, the gradient direction is simply reversed. The behavior of the gradient itself remains the same as when $A^{(i)} > 0$. In the case of $i = y$, RA2C decreases the probability $\pi^{(y)}$ more when $\pi^{(y)}$ is around 0.5. For $i \neq y$, RA2C helps increase $\pi^{(y)}$ more when both $\pi^{(i)}$ and $\pi^{(y)}$ are ambiguous (both around 0.5).

B.2 Symmetric PPO gradient analysis

$$L_{\text{sppo}} = L_{\text{ppo}} + L_{\text{rppo}} \tag{35}$$

For $i = y$ and $A^{(i)} > 0$,

$$\begin{aligned}\frac{\partial L_{\text{sppo}}}{\partial z_y} &= \frac{\partial L_{\text{ppo}}}{\partial z_y} + \frac{\partial L_{\text{rppo}}}{\partial z_y} \\ &= \underbrace{\frac{A^{(i)}\pi^{(i)}(\pi^{(i)} - 1)}{\pi^{(i)}_{\text{old}}}}_{\nabla L_{\text{ppo}} < 0} \underbrace{-\frac{A^{(i)} Z \pi^{(y)}(\pi^{(y)} - 1)}{\pi^{(i)}_{\text{old}}}}_{\nabla L_{\text{rppo}} < 0} \quad \text{by (27) and (29)}\end{aligned} \tag{36}$$

For $i \neq y$ and $A^{(i)} > 0$,

$$\begin{aligned}\frac{\partial L_{\text{sppo}}}{\partial z_y} &= \frac{\partial L_{\text{ppo}}}{\partial z_y} + \frac{\partial L_{\text{rppo}}}{\partial z_y} \\ &= \underbrace{\frac{A^{(i)}\pi^{(i)}\pi^{(y)}}{\pi^{(i)}_{\text{old}}}}_{\nabla L_{\text{ppo}} > 0} \underbrace{-\frac{A^{(i)} Z \pi^{(y)}\pi^{(i)}}{\pi^{(i)}_{\text{old}}}}_{\nabla L_{\text{rppo}} > 0} \quad \text{by (28) and (29)}\end{aligned} \tag{37}$$

For $i = y$ and $A^{(i)} < 0$,

$$\begin{aligned}\frac{\partial L_{\text{sppo}}}{\partial z_y} &= \frac{\partial L_{\text{ppo}}}{\partial z_y} + \frac{\partial L_{\text{rppo}}}{\partial z_y} \\ &= \underbrace{\frac{A^{(i)}\pi^{(i)}(\pi^{(i)} - 1)}{\pi^{(i)}_{\text{old}}}}_{\nabla L_{\text{ppo}} < 0} + \underbrace{\frac{A^{(i)} Z \pi^{(y)}(\pi^{(y)} - 1)}{\pi^{(i)}_{\text{old}}}}_{\nabla L_{\text{rppo}} < 0} \quad \text{by (27) and (29)}\end{aligned} \tag{38}$$

For $i \neq y$ and $A^{(i)} < 0$,

$$\begin{aligned}\frac{\partial L_{\text{sppo}}}{\partial z_y} &= \frac{\partial L_{\text{ppo}}}{\partial z_y} + \frac{\partial L_{\text{rppo}}}{\partial z_y} \\ &= \underbrace{\frac{A^{(i)}\pi^{(i)}\pi^{(y)}}{\pi_{\text{old}}^{(i)}}}_{\nabla L_{\text{ppo}} > 0} + \underbrace{\frac{A^{(i)}Z\pi^{(y)}\pi^{(i)}}{\pi_{\text{old}}^{(i)}}}_{\nabla L_{\text{rppo}} > 0} \quad \text{by (28) and (29)}\end{aligned} \tag{39}$$

Basically, the mechanism of RPPO is the same as RA2C, except for $\pi_{\text{old}}^{(i)}$, which does not change the gradient sign. Therefore, RPPO also helps PPO deviate from ambiguous predictions, acting as an accelerator.

C Experimental setups and results

C.1 Hyperparameters

Atari games: We primarily follow the hyperparameter settings of RL Baselines3 Zoo (Raffin, 2020). Most hyperparameter values remain unchanged across environments. Only α and β are adjusted for the reverse RL loss. For SA2C without noise, we use $(\alpha = 0.5, \beta = 5.0)$ for all environments. For SA2C with noise, we use $(\alpha = 0.5, \beta = 1.0)$ for (Alien, MsPacman, Qbert, TimePilot, VideoPinball, Assault, Gravitar, StarGunner, UpNDown), and $(\alpha = 0.5, \beta = 1.0)$ for others. For SPPO without noise, we use $(\alpha = 0.5, \beta = 1.0)$ for all environments. For SPPO with noise, we use $(\alpha = 0.5, \beta = 10.0)$ for all environments. We do not use any GPU for Atari games.

Table 5: Hyperparameters for Atari games

	Without Noise	$\epsilon \sim$ **BSC**(0.1)
SA2C		
- $(\alpha = 0.5, \beta = 1.0)$	-	(Alien, Assault, Gravitar, MsPacman, Qbert, StarGunner, TimePilot, UpNDown, VideoPinball)
- $(\alpha = 0.5, \beta = 5.0)$	All environments	All others except those mentioned above
SPPO		
- $(\alpha = 0.5, \beta = 1.0)$	All environments	-
- $(\alpha = 0.5, \beta = 10.0)$	-	All environments

MuJoCo and Box2D: We use n_envs $= 4$ and n_steps $= 8$ for A2C and SA2C. We follow Stable-Baselines3's default hyperparameters (Raffin et al., 2021) for other settings. Only α and β are adjusted for the reverse RL loss. For table visibility, let {Ant = 1, BipedalWalker = 2, HalfCheetah = 3, Hopper = 4, HumanoidStandup = 5, InvertedDoublePendulum = 6, LunarLanderContinuous = 7, Swimmer = 8, Walker2d = 9}. We do not use any GPU for these tasks.

Table 6: Hyperparameters for MuJoCO and Box2D environments

	Without Noise	$\epsilon \sim \mathcal{N}(0, 0.05^2)$
SA2C		
- $(\alpha = 0.5, \beta = 0.2)$	(1, 4, 8, 9)	(1)
- $(\alpha = 0.5, \beta = 0.5)$	(2, 5)	-
- $(\alpha = 0.5, \beta = 5.0)$	(3, 6, 7)	(7)
- $(\alpha = 0.5, \beta = 10.0)$	-	(2, 3, 4, 5, 6, 8, 9)
- $Z = -1$	All environments	All environments
- (timesteps= $2e6$)	All environments	All environments
- (Number of bins= 11)	All environments	All environments
SPPO		
- $(\alpha = 0.5, \beta = 20.0)$	All environments	(1, 7)
- $(\alpha = 0.5, \beta = 25.0)$	-	(2, 3, 5, 6, 9)
- $(\alpha = 0.5, \beta = 50.0)$	-	(4, 8)
- $Z = -1$	All environments	All environments
- (timesteps= $1e6$)	(2, 6)	(2, 6)
- (timesteps= $2e6$)	(1, 3, 4, 8, 9)	(1, 3, 4, 8, 9)
- (timesteps= $5e6$)	(9)	(9)
- (timesteps= $1e7$)	(5)	(5)
- (Number of bins= 11)	All environments	All environments

IMDB and TL;DR: We basically use the provided implementation (Chang et al., 2023) and follow their hyperparameters, with the addition of the advantage normalization step for PPO. The scripts used in our experiments are available in the code repository for further detail. We use a single Nvidia A100 (80GB) for our experiments.

Table 7: Hyperparameters for IMDB positive sentiment and TL;DR summarization

	IMDB	**Box2D**
PPO		
- model:	GPT-2	GPT-J
- updates:	60	100
- trajectories per update:	112	64
- epochs per update	5	4
- batch size	28	32
- learning rate	5e-6	5e-6
- discount factor	0.99	1.0
- GAE lambda	0.95	0.95
- clip range	0.2	0.2
SPPO	$(\alpha = 0.5, \beta = 0.4)$	$(\alpha = 0.5, \beta = 0.2)$

C.2 Experimental results: A2C and SA2C

Table 8: Mean final scores and standard errors (over the last 10 episodes) of A2C and SA2C on Atari games, without and with binary symmetric channel (BSC) noise with a crossover probability of 0.1 across 5 seeds.

	Without Noise		$\epsilon \sim$ **BSC**(0.1)	
	A2C	**SA2C**	A2C	**SA2C**
Alien	**913 ± 100**	771 ± 51	481 ± 72	**496 ± 37**
Assault	**1538 ± 199**	1061 ± 41	287 ± 226	**399 ± 133**
Asterix	2308 ± 86	**2377 ± 164**	1403 ± 305	**1430 ± 208**
BeamRider	1121 ± 61	**1335 ± 43**	**1087 ± 339**	902 ± 196
Centipede	**3588 ± 430**	3574 ± 295	3108 ± 243	**3540 ± 194**
CrazyClimber	98774 ± 2516	**99330 ± 4371**	93042 ± 8711	**97058 ± 6251**
DemonAttack	4309 ± 325	**5017 ± 625**	**30 ± 21**	19 ± 3
Frostbite	255 ± 2	**257 ± 3**	241 ± 9	**286 ± 48**
Gopher	960 ± 80	**1036 ± 138**	947 ± 91	**996 ± 114**
Gravitar	143 ± 18	**201 ± 16**	**279 ± 48**	183 ± 36
Krull	6387 ± 267	**7672 ± 819**	**7564 ± 486**	6337 ± 754
MsPacman	1175 ± 43	**1495 ± 104**	**926 ± 44**	916 ± 100
NameThisGame	**5945 ± 102**	5614 ± 166	2280 ± 257	**2372 ± 141**
Qbert	1646 ± 240	**2103 ± 261**	620 ± 96	**641 ± 77**
Riverraid	4368 ± 582	**5461 ± 456**	1609 ± 65	**2511 ± 190**
RoadRunner	14971 ± 1396	**18624 ± 1812**	**5606 ± 1788**	3830 ± 1517
Seaquest	836 ± 7	**988 ± 92**	650 ± 22	**653 ± 22**
StarGunner	**2222 ± 114**	1766 ± 120	**1194 ± 645**	622 ± 54
TimePilot	**3992 ± 198**	3116 ± 137	2232 ± 259	**3288 ± 106**
UpNDown	**8313 ± 1544**	1638 ± 761	4228 ± 1187	**7093 ± 2772**
VideoPinball	**24948 ± 3038**	19618 ± 1888	20319 ± 2157	**25035 ± 3914**
WizardOfWor	**824 ± 136**	674 ± 125	496 ± 87	**752 ± 156**
Wins (SA2C)	**12 / 22**		**15 / 22**	

Table 9: Mean final scores and standard errors (over the last 10 episodes) of A2C and SA2C on MuJoCo benchmark tasks and Box2D environments without Gaussian noise across 30 seeds.

Without Noise	**Ant**	**Hopper**	**HalfCheetah**	**HumanoidStandup**
A2C	757 ± 116	1410 ± 112	1393 ± 163	121850 ± 4264
DA2C	2220 ± 96	$\mathbf{1944 \pm 116}$	$\mathbf{2325 \pm 209}$	152135 ± 3937
DSA2C	$\mathbf{2287 \pm 94}$	1797 ± 139	2266 ± 203	$\mathbf{159142 \pm 129}$
	Walker2d	**Swimmer**	**BipedalWalker**	**LunarLanderContinuous**
A2C	1348 ± 130	95.8 ± 19.0	124 ± 23	79.0 ± 20.2
DA2C	$\mathbf{2131 \pm 154}$	$\mathbf{142.4 \pm 17.0}$	234 ± 22	176.7 ± 20.9
DSA2C	1662 ± 164	128.5 ± 16.2	$\mathbf{274 \pm 16}$	$\mathbf{221.2 \pm 10.7}$
	InvertedDoublePendulum			
A2C	1670 ± 500			
DA2C	9139 ± 94			
DSA2C	$\mathbf{9145 \pm 93}$			

Table 10: Mean final scores and standard errors (over the last 10 episodes) of A2C and SA2C on MuJoCo benchmark tasks and Box2D environments with Gaussian noise (mean 0 and standard deviation 0.05) across 30 seeds.

$\epsilon \sim \mathcal{N}(0, 0.05^2)$	**Ant**	**Hopper**	**HalfCheetah**	**HumanoidStandup**
A2C	673 ± 108	1083 ± 92	1610 ± 163	101064 ± 4933
DA2C	1296 ± 80	$\mathbf{1323 \pm 87}$	1510 ± 126	126241 ± 3973
DSA2C	$\mathbf{1520 \pm 83}$	1307 ± 102	$\mathbf{1696 \pm 163}$	$\mathbf{128064 \pm 4391}$
	Walker2d	**Swimmer**	**BipedalWalker**	**LunarLanderContinuous**
A2C	786 ± 86	28.9 ± 4.4	158 ± 20	-3.7 ± 15.9
DA2C	$\mathbf{1599 \pm 138}$	36.8 ± 4.6	210 ± 21	106 ± 20
DSA2C	1423 ± 129	$\mathbf{53.1 \pm 7.0}$	$\mathbf{222 \pm 20}$	$\mathbf{179 \pm 12}$
	InvertedDoublePendulum			
A2C	3852 ± 634			
DA2C	7900 ± 364			
DSA2C	$\mathbf{8323 \pm 217}$			

C.3 EXPERIMENTAL RESULTS: PPO AND SPPO

Table 11: Mean final scores and standard errors (over the last 10 episodes) of PPO and SPPO on Atari games, without and with binary symmetric channel (BSC) noise with a crossover probability of 0.1 across 5 seeds.

	Without Noise		$\epsilon \sim$ **BSC**(0.1)	
	PPO	**SPPO**	PPO	**SPPO**
Alien	**1128 ± 105**	1081 ± 79	525 ± 26	**713 ± 26**
Assault	3134 ± 193	**3385 ± 214**	2327 ± 401	**3698 ± 363**
Asterix	2599 ± 101	**2976 ± 150**	1272 ± 106	**1739 ± 329**
BeamRider	**2176 ± 251**	1635 ± 404	**1828 ± 130**	1580 ± 96
Centipede	2961 ± 379	**3694 ± 224**	4759 ± 257	**7525 ± 769**
CrazyClimber	86764 ± 3568	**103588 ± 2871**	71144 ± 11060	**99810 ± 2487**
DemonAttack	7872 ± 302	**7901 ± 455**	**161 ± 24**	132 ± 13
Frostbite	268 ± 5	**286 ± 6**	**509 ± 108**	23 ± 16
Gopher	787 ± 48	**875 ± 78**	478 ± 38	**7765 ± 3366**
Gravitar	371 ± 47	**442 ± 67**	269 ± 39	**332 ± 61**
Krull	6628 ± 417	**7578 ± 588**	5602 ± 481	**9015 ± 381**
MsPacman	837 ± 62	**1204 ± 86**	704 ± 41	**1011 ± 52**
NameThisGame	**5665 ± 280**	5423 ± 63	2681 ± 143	**5187 ± 247**
Qbert	4352 ± 128	**4412 ± 282**	2827 ± 1927	**4020 ± 2415**
Riverraid	6128 ± 272	**6343 ± 219**	2460 ± 127	**3998 ± 248**
RoadRunner	**28382 ± 2254**	22562 ± 2875	1204 ± 157	**3830 ± 1230**
Seaquest	**902 ± 2**	888 ± 6	652 ± 16	**814 ± 15**
StarGunner	11848 ± 722	**14746 ± 1876**	1514 ± 110	**23250 ± 6292**
TimePilot	**3850 ± 151**	3548 ± 220	3506 ± 318	**3936 ± 420**
UpNDown	58289 ± 21226	**126830 ± 27534**	8815 ± 1395	**73490 ± 33553**
VideoPinball	22408 ± 4292	**29485 ± 2851**	31680 ± 2318	**37048 ± 6989**
WizardOfWor	3186 ± 256	**3762 ± 387**	940 ± 158	**4442 ± 1332**
Wins (SPPO)	**16 / 22**		**19 / 22**	

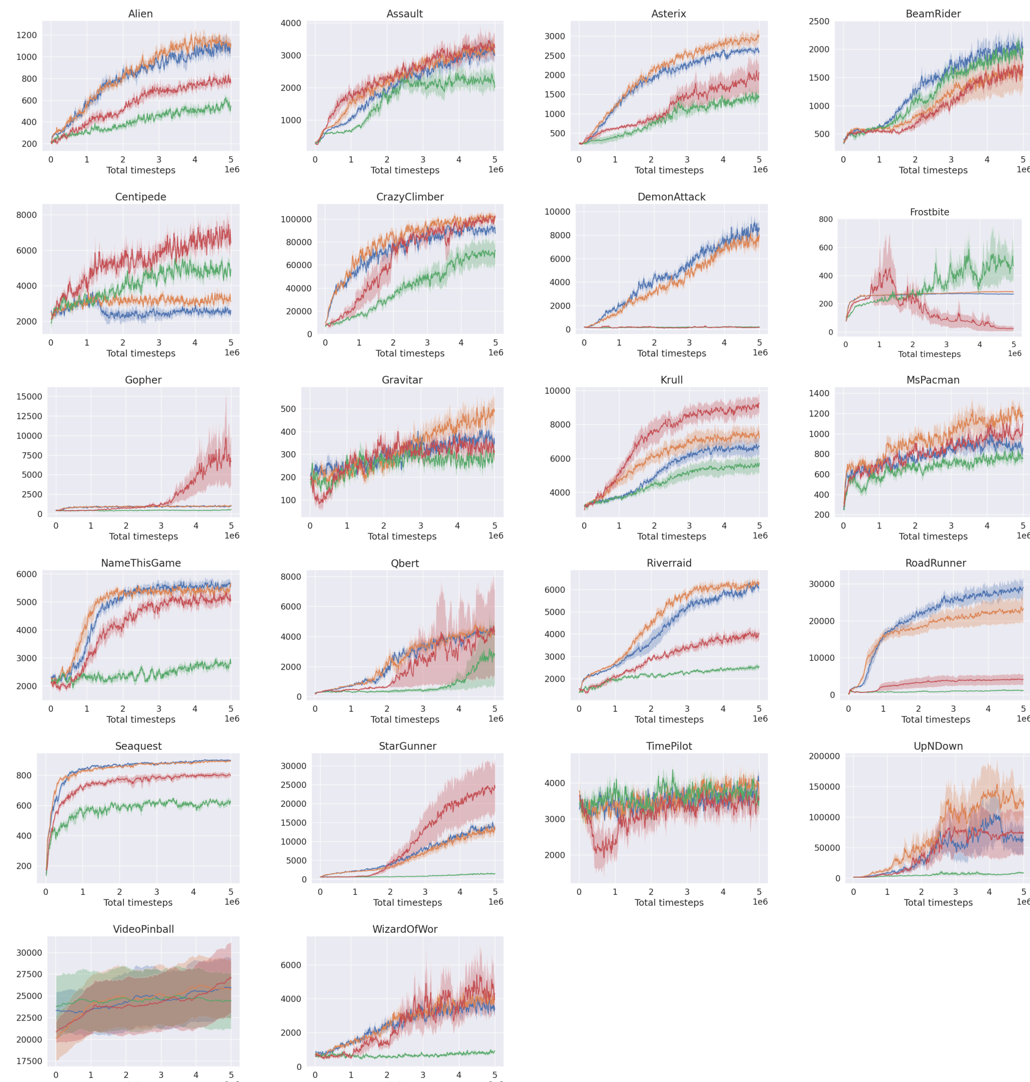

Figure 3: Result of training plots for SPPO and PPO for Atari games. The blue line indicates the original PPO without any added noise, while the orange line represents SPPO without added noise. The green line indicates PPO with 10% noise, and the red line represents SPPO with 10% noise. We fix $\alpha = 0.5$ for all environments, with $\beta = 1.0$ for the experiments without noise and $\beta = 10.0$ for the noise environments.

Table 12: Mean final scores and standard errors (over the last 10 episodes) of PPO and SPPO on MuJoCo benchmark tasks and Box2D environments without Gaussian noise across 30 seeds.

Without Noise	**Ant**	**Hopper**	**HalfCheetah**	**HumanoidStandup**
PPO	2068 ± 166	$\mathbf{2875 \pm 137}$	2282 ± 191	93763 ± 3402
DPPO	2735 ± 109	2154 ± 119	3478 ± 279	176320 ± 6538
DSPPO	$\mathbf{2885 \pm 100}$	2299 ± 115	$\mathbf{4104 \pm 258}$	$\mathbf{189301 \pm 5915}$
	Walker2d	**Swimmer**	**BipedalWalker**	**LunarLanderContinuous**
PPO	2793 ± 199	112 ± 5.0	247 ± 8.3	134 ± 10.9
DPPO	4443 ± 119	$\mathbf{131 \pm 0.3}$	265 ± 15.5	241 ± 7.7
DSPPO	$\mathbf{4587 \pm 154}$	130 ± 0.6	$\mathbf{274 \pm 6.2}$	$\mathbf{250 \pm 6.9}$
	InvertedDoublePendulum			
PPO	7454 ± 394			
DPPO	8928 ± 136			
DSPPO	$\mathbf{9015 \pm 101}$			

Table 13: Mean final scores and standard errors (over the last 10 episodes) of PPO and SPPO on MuJoCo benchmark tasks and Box2D environments with Gaussian noise (mean 0 and standard deviation 0.05) across 30 seeds.

$\epsilon \sim \mathcal{N}(0, 0.05^2)$	**Ant**	**Hopper**	**HalfCheetah**	**HumanoidStandup**
PPO	601 ± 47	1936 ± 147	2068 ± 208	80945 ± 2130
DPPO	1897 ± 86	2153 ± 106	2722 ± 188	$\mathbf{146038 \pm 1841}$
DSPPO	$\mathbf{2095 \pm 102}$	$\mathbf{2333 \pm 109}$	$\mathbf{3118 \pm 195}$	145974 ± 2520
	Walker2d	**Swimmer**	**BipedalWalker**	**LunarLanderContinuous**
PPO	1270 ± 107	44 ± 3.0	158 ± 15.2	181 ± 13.8
DPPO	3419 ± 100	57 ± 3.6	$\mathbf{274 \pm 7.1}$	281 ± 5.7
DSPPO	$\mathbf{3523 \pm 129}$	$\mathbf{72 \pm 5.1}$	267 ± 8.8	$\mathbf{294 \pm 3.3}$
	InvertedDoublePendulum			
PPO	8050 ± 244			
DPPO	8963 ± 100			
DSPPO	$\mathbf{9147 \pm 61}$			

C.4 On and off advantage normalization

Table 14: Comparison with and without advantage normalization over 4 different random seeds.

PPO	IMDB	TL;DR
Without A Normalization	0.77 ± 0.01	6.06 ± 0.02
With A Normalization	0.89 ± 0.02	5.94 ± 0.08

D Examples of reward model errors

Warning: This section contains harmful language.

Table 15: Example showing a trained reward model with errors that are not consistent for empty outputs, and the reward for an empty output is greater than that for a non-empty summarization. [...] indicates omitted content for brevity.

Subreddit: r/relationships (Sample ID: 37) **TITLE:** I'm a dumb [21] male and so I'm having a lot of trouble interpreting the signals that this [21] girl may or may not be sending me. A little help please? **Post:** So okay, I'm from New York but I study in Oregon for most of the year. Recently a friend of mine who I was not really close started facebook messaging me, that was about 3 months ago, since then we've talked almost everyday. [...] Is she trying to play hard to get? Am I looking way too into this and maybe she was just occupied that weekend? I really have no idea how to evaluate this. Do any of you guys have any suggestions/ideas?
Generated Summary: <empty>
Reward Model Output: 6.66
Subreddit: r/relationship_advice (Sample ID: 60) **TITLE:** My bf [23] doesn't speak of his childhood, but I[f22] know he's traumatized. **Post:** We were friends for 10 years, before we got together. He than told me once about his terrible childhood. (He told only 3 of his friends his story) Now we're a couple for quite a few months and well, sometimes there's stuff I know that reminds him of his childhood, but it's like he's forgotten that he had told me. [...] (But striking, the things he thinks are important are always the things his parents should have done, to save him from the traumatizing stuff.)I know he likes to put his problems far away. But on the other hand, I'm his girlfriend now and we're pretty serious, isn't it good to speak about it maybe just once, so he knows I know his secret/won't tell, and most of all, I'm always there for him? What do you think?
Generated Summary: <empty>
Reward Model Output: 3.14
Subreddit: r/AskReddit (Sample ID: 27) **TITLE:** Dear Reddit, What silly/irrelevant/rediculous family miscommunications have lead to feuds lasting years? **Post:** My Grandma and my aunt (her daugher-in-aw) haven't spoken to each other in years over a phone that didn't get hung up. My aunt and uncle screen their calls and frequently do not return them– one time, my grandma called and left a message then thought she hung up the phone. [...] Why continue to hold a silly grudge? To complicate matters further, my grandma has a daughter who lives with her and likes to be in other peoples business– I think she is also part of the problem here as she won't drop it either. Grandma is innocent but has a daughter and daughter-in-law who won't grow up and drop it
Generated Summary: Grandma and Aunt haven't spoken in years over a phone that didn't get hung up. Grandma wants to reconcile and clear the air, but Aunt won't go near her, won't let her husband and kids go there, and avoids.
Reward Model Output: 5.40