Navigating Noisy Feedback: Enhancing Reinforcement Learning with Error-Prone Language Models

Muhan Lin[1], Shuyang Shi[1], Yue Guo[1], Behdad Chalaki[2], Vaishnav Tadiparthi[2], Ehsan Moradi Pari[2], Simon Stepputtis[1], Joseph Campbell [1], Katia Sycara[1]

[1]Carnegie Mellon University, [2]Honda Research Institute USA

Correspondence: muhanlin@cs.cmu.edu

Abstract

The correct specification of reward models is a well-known challenge in reinforcement learning. Hand-crafted reward functions often lead to inefficient or suboptimal policies and may not be aligned with user values. Reinforcement learning from human feedback is a successful technique that can mitigate such issues, however, the collection of human feedback can be laborious. Recent works have solicited feedback from pre-trained large language models rather than humans to reduce or eliminate human effort, however, these approaches yield poor performance in the presence of hallucination and other errors. This paper studies the advantages and limitations of reinforcement learning from large language model feedback and proposes a simple yet effective method for soliciting and applying feedback as a potential-based shaping function. We theoretically show that inconsistent rankings – which approximate ranking errors – lead to uninformative rewards with our approach. Our method empirically improves convergence speed and policy returns over commonly used baselines even with significant ranking errors, and eliminates the need for complex post-processing of reward functions.

1 Introduction

The correct specification of task rewards is a well-known challenge in reinforcement learning (RL) (Leike et al., 2018). Complex tasks often necessitate complex, nuanced reward models, particularly as shaping terms may be required to guide exploration. However, hand-crafting these reward functions is difficult and often leads to a phenomenon known as *reward hacking*, wherein an agent learns to exploit a reward function for increased returns while yielding unexpected or undesired behavior (Skalse et al., 2022). Reward hacking is symptomatic of the broader challenge of *value alignment*, in which it is difficult for a human domain expert to fully and accurately specify the desired behavior of the learned policy in terms of a reward function.

In this study, we aim to eliminate the dependence on handcrafted reward functions by training agents with reward functions derived from data. A notable method in this domain is reinforcement learning from human feedback (RLHF), where policy trajectories are ranked by humans. These rankings are then used to learn a reward function which guides model training and facilitates value alignment. This process is extremely costly in terms of human effort, however, requiring a significant number of rankings to train accurate reward models (Casper et al., 2023).

We can avoid the need for humans-in-the-loop by instead generating rankings with pre-trained large language models (LLMs) in a process known as reinforcement learning with AI feedback (RLAIF) (Lee et al., 2023; Bai et al., 2022; Kim et al., 2023). However, LLMs are well known to hallucinate and present false information as fact (Zhang et al., 2023), which reduces the accuracy and reliability of the resulting rankings. This is often overcome through complex reward post-processing techniques, which may be task-specific and difficult to tune (Klissarov et al., 2023).

In this work, we propose a simple and effective strategy for reinforcement learning in the face of unreliable LLM feedback. The core idea underlying our approach is to issue uninformative rewards for states in which the LLM is uncertain. Thus, we avoid issuing potentially misleading rewards which allows us to train performant policies even in the face of significant ranking errors. Building off the insight that certainty in language models is expressed through output consistency (Tanneru et al., 2024), we show that rewards issued from a potential-based scoring function learned over repeated rankings naturally reflect an LLM's uncertainty.

Our contributions are as follow, we 1) introduce

Findings of the Association for Computational Linguistics: EACL 2024, pages 16002–16014

a methodology for incorporating noisy LLM feedback into RL which out-performs prior SOTA; and 2) provide theoretical and empirical analysis showing that uncertain LLM outputs – as given by inconsistent responses – lead to uninformative rewards which improve convergence speed and policy returns in experiments. The codes of this work can be accessed here.

2 Related Works

Constructing rewards based on human feedback has a long history (Thomaz et al., 2006). To efficiently use human domain knowledge and provide more generalizable rewards, human preference on episode segments (Sadigh et al., 2017; Christiano et al., 2017; Bıyık et al., 2019) and human demonstrations (Bıyık et al., 2022) are distilled into models which serve as reward functions for RL. The method has witnessed great success in complex domains where rewards are difficult to specify such as training large language models (LLM) to align with human logic and common sense (Ziegler et al., 2019; Ouyang et al., 2022).

One major drawback of RLHF is its requirement of exhaustive human participation to provide demonstrations and feedback. LLMs have shown deductive logic abilities comparable to humans in recent years (Du et al., 2023), and are able to substitute humans in reward issuing (Kwon et al., 2023; Yu et al., 2023; Lee et al., 2023; Xie et al., 2023), or data collection and labeling for reward model training (Lee et al., 2023; Klissarov et al., 2023). While the former suffers from time and resource costs for training RL agents, the latter is becoming promising for training complex RL tasks (Wang et al., 2024).

An outstanding challenge with leveraging LLM-based feedback is that the performance of RLHF is dependent on the quality of feedback received (Casper et al., 2023). Different LLMs have distinct probabilities of giving wrong feedback, thus leading to rewards of varying quality. Casper et al. (2023) also suggests that comparison-based feedback may not be efficient and adequate to train reward models with noisy LLM outputs. In this work, we analyze the training performance of reinforcement learning agents across various LLMs, each of which produce different error distributions in feedback.

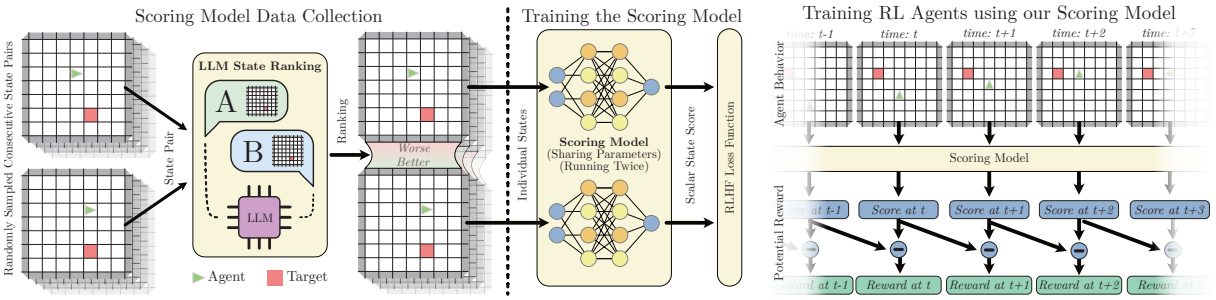
Another challenge is that of the reward formulation itself. Many works train a model distilling LLM or human preference and use it as the reward model (Christiano et al., 2017; Wang et al., 2024; Klissarov et al., 2023; Lee et al., 2023), but in practice, this needs post-processing on outputs of the reward model such as filtering (Klissarov et al., 2023), and normalization (Christiano et al., 2017). Our work posits that a reward function trained without complex post-processing and environment rewards would be more general and adaptable to various practical scenarios.

3 Background

Reinforcement Learning: In reinforcement learning, an agent interacts with an environment and receives a reward for its current action at each time step, learning an optimal action policy to maximize the rewards over time. This procedure can be formulated as an infinite horizon discounted Markov Decision Process (MDP) (Sutton and Barto, 2018).

At each discrete timestep t in this process, the agent observes environment state s_t and takes action a_t, leading to the next environment state s_{t+1} and a reward r_t. An MDP is represented as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma)$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{R} : \mathcal{S} \mapsto \mathbb{R}$ is a reward function, $\mathcal{T}(s, a, s') = P(s'|s, a)$ is a transition function, and γ is a discount factor. A stochastic policy $\pi(a|s) : \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ indicates the probability that the agent selects action a given the state s. The agent's goal is to learn π maximizing the expected discounted return through training, given an initial state distribution.

Preference-based Reinforcement Learning: Our work is based on the framework of preference-based reinforcement learning, where the reward function is learned from preference labels over agent behaviors (Christiano et al., 2017; Ibarz et al., 2018; Lee et al., 2021a,b). For a pair of states (s_a, s_b), an annotator gives a preference label y that indicates which state is ranked higher, considering which state is closer to the given task goal. The preference label $y \in \{0, 1\}$, where 0 indicates s_a is ranked higher than s_b, and 1 indicates s_b is ranked higher than s_a. Given a parameterized state-score function σ_ψ, which is commonly called the potential function and usually equated with the reward model r_ψ, we compute the preference probability of a state pair with the standard Bradley-Terry



(a) Training the scoring model from LLM-ranked state pairs.

(b) Agent training using the scoring model.

Figure 1: Our approach: (a) We train our scoring model with randomly sampled *consecutive* state pairs, which are ranked by an LLM with respect to task completion. The resulting dataset of ranked state pairs is utilized in an RLHF fashion to train a single scoring model, capable of providing a score for any novel state. (b) Using the scoring model, an RL agent is trained by scoring each state. Prior work uses this score as a reward; however, our approach utilizes the score differences as a potential reward.

model (Bradley and Terry, 1952),

$$P_\psi[s_b \succ s_a] = \frac{\exp\left(\sigma_\psi(s_b)\right)}{\exp\left(\sigma_\psi(s_a)\right) + \exp\left(\sigma_\psi(s_b)\right)} = sigmoid(\sigma_\psi(s_b) - \sigma_\psi(s_a)), \quad (1)$$

where $s_b \succ s_a$ indicates s_b is ranked higher than the state s_a. With a preference dataset $D = (s_a^i, s_b^i, y^i)$, preference-based RL learns the state-score model σ_ψ by minimizing the cross-entropy loss, which aims to maximize the score difference between the high and low states:

$$\mathcal{L} = -\mathbb{E}_{(s_a,s_b,y)\sim\mathcal{D}}\Big[\mathbb{I}\{y = (s_a \succ s_b)\} \log P_\psi[s_a \succ s_b] + \mathbb{I}\{y = (s_b \succ s_a)\} \log P_\psi[s_b \succ s_a]\Big]. \quad (2)$$

This framework is used in both RLHF and RLAIF where rewards are issued directly from the state-score model and differ only in the choice of annotator, i.e., human or LLM.

4 Methodology

Despite the success of LLMs in few-shot task generalization, these models are imperfect and yield sub-optimal performance in many areas. One notable issue is the well-documented tendency of LLMs to hallucinate, which results in LLM-generated preference rankings frequently containing errors (see Table 1). These errors present major challenges for reinforcement learning from LLM feedback, as they result in noise in the learned score function. Under the standard RLHF formulation where rewards are directly issued from the score function (Christiano et al., 2017), this can lead to inefficient exploration at best and, at worst, trap the agent in sub-optimal local minima.

4.1 Quantifying Feedback Error through Output Consistency

It has been shown that the certainty of LLM predictions can be quantified by issuing the same query multiple times and measuring the *consistency* of the predictions (Lyu et al., 2024). Specifically, the confidence of ranking s_a higher than s_b, $conf\{y = (s_a \succ s_b)\}$, is defined as $\frac{N(s_a \succ s_b)}{N_{query}(s_a,s_b)}$. where $N(s_a \succ s_b)$ denotes the number of times LLM ranks s_a higher than s_b, and $N_{query}(s_a, s_b)$ denotes the total number of queries on s_a and s_b. Confidence is a necessary condition to consider when evaluating LLM feedback quality, given that low confidence often causes considerable noise in feedback which manifests as incorrect rewards. Based on the definition of feedback confidence, we derive an equivalent form of the RLHF loss based on ranking confidence and consistency as follows:

$$\begin{aligned}\mathcal{L} &= -\mathbb{E}_{(s_a,s_b,y)\sim\mathcal{D}}\Big[\mathbb{E}_{N_{query}}\Big[\mathbb{I}\{y = (s_a \succ s_b)\} \log P_\psi[s_a \succ s_b] + \mathbb{I}\{y = (s_b \succ s_a)\} \log P_\psi[s_b \succ s_a]\Big]\Big] \\ &= -\mathbb{E}_{(s_a,s_b,y)\sim\mathcal{D}}\Big[conf\{y = (s_a \succ s_b)\} \log(sigmoid(\sigma_\psi(s_a) - \sigma_\psi(s_b))) + conf\{y = (s_b \succ s_a)\} \log(sigmoid(\sigma_\psi(s_b) - \sigma_\psi(s_a)))\Big]. \end{aligned} \quad (3)$$

This loss function uses confidence-based weights to relate the scores between each state in ranked pairs. From this derivation, we see the following. 1) A more confident ranking produces a larger score difference between the ranked states, i.e., the magnitude of the score difference is proportional to the confidence. Formally, if $conf\{y = (s_a \succ s_b)\} > conf\{y = (s_b \succ s_a)\}$ then $\sigma_\psi(s_a) - \sigma_\psi(s_b) > 0$. In the event the LLM is perfectly confident, $conf\{y = (s_a \succ s_b)\} = 1$, then the loss function will maximize $\sigma_\psi(s_a) - \sigma_\psi(s_b)$. 2) As the confidence *decreases*, then $|conf\{y = (s_a \succ s_b)\} - conf\{y = (s_b \succ s_a)\}|$ converges to 0. When the LLM is completely uncertain then $conf\{y = (s_a \succ s_b)\} = conf\{y = (s_b \succ s_a)\}$ and the loss function will minimize $|\sigma_\psi(s_a) - \sigma_\psi(s_b)|$, resulting in identical state scores such that $\sigma_\psi(s_a) = \sigma_\psi(s_b)$. The formal proof is given in Appendix A.

4.2 Potential-based Rewards for Learning with Noisy Feedback

The above observations stemming from Eq. 5 motivate the form of our proposed method. Intuitively, when the LLM is completely uncertain when ranking s_a and s_b then the difference between their scores is zero. This is ideal, as **when the LLM is unable to issue an accurate ranking then we would like it to issue an uninformative reward**, i.e., a reward of zero. Our solution is to treat the state-score as a *potential function*, defining the reward as the difference between the scores of successive state pairs:

$$r(s_t) = \sigma_\psi(s_t) - \sigma_\psi(s_{t-1}). \tag{4}$$

Thus, the more uncertain an LLM's ranking is, the less informative the reward is. The potential in Eq. 4 is naturally shaped to a proper scale range, with positive rewards for actions that are beneficial and promising to the given task goal and negative rewards for detrimental actions. Large values correspond to more confident rankings, while small ones to less confident rankings. As such, our approach is particularly well-suited to RLAIF with smaller, specialized models which are often necessary in resource-constrained environments.

There is an additional benefit to this formulation. Prior works treat the state-score function as a reward function and directly issue rewards from it, which we call the "direct-reward" method. This often leads to training instability as the rewards may have significant differences in scale, which need to be corrected via post-processing techniques such as normalization and thresholding as well as extrinsic per-step reward penalties. However, the performance of post-processed direct rewards is highly sensitive to these hyper-parameters, as they are often task-specific. Our potential difference formulation helps alleviate this issue as 1) uncertain states converge to the same score value so the impact of noisy rankings no longer needs to be mitigated through post-processing, and 2) per-step penalties can be discarded in favor of simple timestep-based discounting which are far less sensitive.

4.3 Algorithm

Our algorithm consists of the following four steps: 1) Randomly sample pairs of sequential states from the environment. 2) Query the LLM to rank states in each pair with respect to a natural language task description, e.g., "Go to the green goal". The prompt contains a language task description, environment description, and in-context learning examples (Wei et al., 2022) as context to generate preference labels for states in each pair. 3) Train the state-score model σ_ψ with the loss function in Eq. 2. 4) Train a reinforcement learning agent with feedback from the state-score model.

5 Performance Analysis of Potential-Difference Rewards

We evaluate our approach in commonly-used discrete (Grid World) and continuous (MuJoCo) (Brockman et al., 2016) benchmark environments. Throughout these experiments, we investigate the effectiveness of our potential-based reward function a) as compared to using the score as a reward directly in both single and multi-query settings; and b) its sensitivity to inconsistency in state rankings.

5.1 Experiment Setup

Grid World. We examine three scenarios within Grid World (Swamy et al., 2024): **NoLock**, **Lock**, and **MultiLock**. The layouts are shown in Fig. 2. In each scenario, the agent (green triangle) must navigate to the target (green rectangle). There are one and two locked doors in the Lock and MultiLock variants, respectively, that block the agent's way to the goal. To unlock each door the agent must pick up the appropriate key and use it on the door. The agent, goal, and key positions are randomly initialized in every episode.

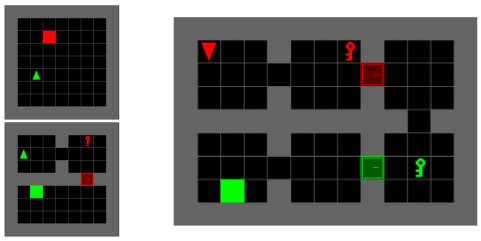


Figure 2: Grid world environments with NoLock (upper-left), Lock (lower-left), and MultiLock (right) variants from left to right.

MuJoCo. We examine a subset of robot control tasks selected from the simulated robotics benchmark MuJoCo (Todorov et al., 2012). We choose 3 tasks with increasing degrees of complexity: **Inverted Pendulum**, **Reacher**, and **Hopper**.

For each of these six environments, we compare our approach with the following baselines:

- **Direct reward** directly utilizes the trained state-score functions' score as reward; i.e., $r(s) = \sigma_\psi(s)$. Following Christiano et al. (2017), the reward is normalized to zero-mean with a standard deviation of 1.

- **Default reward** utilizes the vanilla RL objective of each environment with human-specified reward functions. In grid world variants, the default reward is defined as 0 for failure and $1 - 0.99n/n_{max}$ otherwise when the agent reaches the goal, where n_{max} is the maximum time steps for each episode and n is the step count until success. For MuJoCo tasks, the default rewards are specified as those defined in OpenAI Gym (Brockman et al., 2016).

In each environment, we randomly sample pairs of sequential states from the environment with replacement in order to generate rankings for training the state-score model used by both potential difference and direct reward. For single-query experiments, Grid World uses 2500, 3500, and 6000 samples for NoLock, Lock, and MultiLock respectively while MuJoCo uses 1000 samples for each environment.

Without loss of generality, we employ PPO as the underlying policy-training framework (Schulman et al., 2017) and make the following assumptions: a) the environment is fully observable; and b) the agent has no knowledge of the task before training, i.e., is randomly initialized.

Env.	Mthd.	GT	Llama-3 8B	Mixtral	Llama-3 70B	GPT-4
No Lock	Rank	1.0	0.69	0.76	0.93	1.0
	Score	1.0	0.77	0.89	0.98	1.0
Lock	Rank	1.0	0.54	0.65	0.89	0.98
	Score	1.0	0.55	0.74	0.97	0.98
Multi Lock	Rank	1.0	0.58	0.60	0.90	0.99
	Score	1.0	0.66	0.66	0.96	0.99

Table 1: Ranking accuracy for each LLM across 1000 state pairs sampled from each environment. Rank indicates the direct ranking performance of the LLMs and Score indicates the ranking performance of the trained score models. Given that the ground-truth ranking are only accessible in grid world environments, we only show the ranking correctness of LLMs and state-score models in these three environments.

5.2 LLM Ranking Performance

We first quantify the performance of four different LLM models used in this work over each Grid World environment. After sampling pairs of sequential states as discussed in Sec. 5.1, we measure the accuracy of a) the LLM's rankings and b) the resulting state-score model with respect to ground truth rankings. The results, shown in Table 1, provide us with an approximate ordering of LLM ranking performance, where GPT-4 $>$ Llama-3 70B $>$ Mixtral $>$ Llama-3 8B.

5.3 Single-Query Evaluation

We next examine how our approach performs compared to the standard direct reward approach commonly utilized in RLHF. In each environment, we train our state score models with 4 LLMs: Mixtral (Jiang et al., 2024), GPT 4 (Achiam et al., 2023), and Llama-3 with 8B and 70B parameters (Touvron et al., 2023). For Grid World environments, we add an additional baseline in which rankings are generated using a ground truth heuristic function (GT) which serves as an upper bound for our methods.

The state score models are trained by minimizing the loss in Eq. 2. Then they are employed to train 5 RL policies with random seeds and initializations for each method. As a common approach to avoid reward hacking, a constant step penalty is applied to the produced rewards from both methods in all environments except for MuJoCo Reacher, which exploits a torque penalty as described in Brockman et al. (2016). The results, as well as the default reward performance, are shown in Fig. 3 and Fig. 4.

In Grid World environments, our method (in

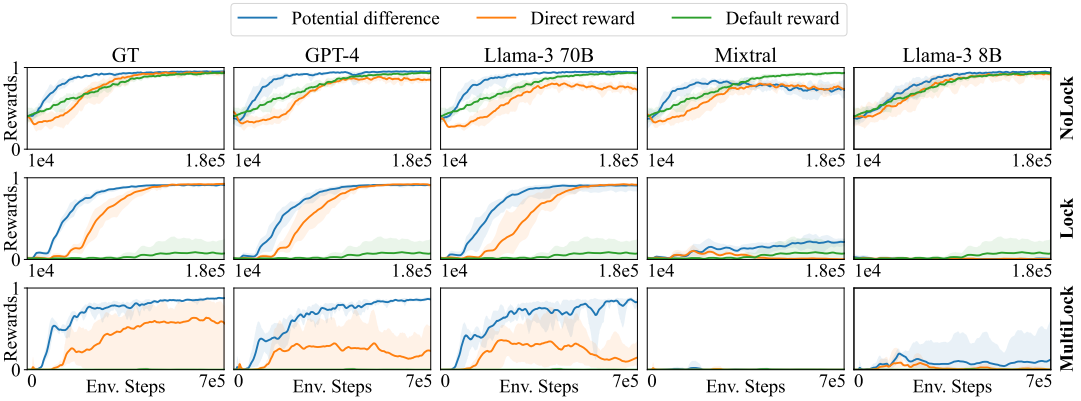


Figure 3: The average learning curves with reward functions trained from single LLM queries in the Grid World environments over 5 random seeds, with the return variance visualized as shaded areas.

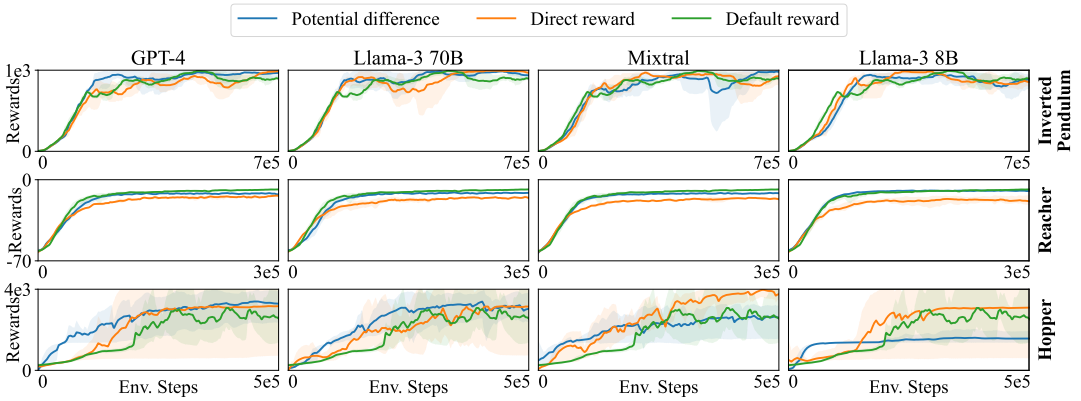


Figure 4: The average learning curves with reward functions trained from single LLM queries in the MuJoCo environments over 5 random seeds, with the return variance visualized as shaded areas.

blue) of potential difference-based reward outperforms the direct reward method in most cases. When using GT, GPT-4, or Llama-3 70B rankings, our method converges the fastest and yields the highest final reward. For the environments of Lock and MultiLock (bottom two rows), the tasks are more challenging when using the default reward; however, our method remains on par, or outperforms the baseline with respect to convergence speed and final reward. However, when using LLMs which generate noisy outputs (i.e., Mixtral and Llama-3 8B), all methods fail to converge in the Lock and MultiLock environments. In Sec. 5.4, we detail our approach of using multiple queries, particularly for low-performing LLMs, to regain training performance using our potential-based reward function.

In MuJoCo environments, reported in Fig. 4, our potential-based reward method slightly outperforms (particularly in Hopper with GPT-4 and Llama-3 70B) or is on par with our baseline methods. Exceptions to this trend can be seen with low-performing LLMs (e.g., Llama-3 8b). Our method outperforms direct reward in Reacher and achieves a performance similar to the well-crafted default reward function, showing that potential-difference rewards are better. However, direct reward outperforms ours when using low-performing LLMs, particularly Mixtral and Llama-3 8B. We attribute this to the challenge of designing appropriate prompts based on human intuition, i.e., we prompt LLMs to compare the hopper's speed in two consecutive states because the hopper should learn to move forward without falling down. However, these LLMs then encourage moving faster instead of simply moving forward. While this prompt could lead to a good reward for high-performing LLMs, low-performing LLMs could not handle such situations, and we hypothesize that this leads to sub-optimal training results.

5.3.1 Hyper-Parameter Sensitivity Analysis

Since potential difference reward and direct reward suffer from reward hacking without post-

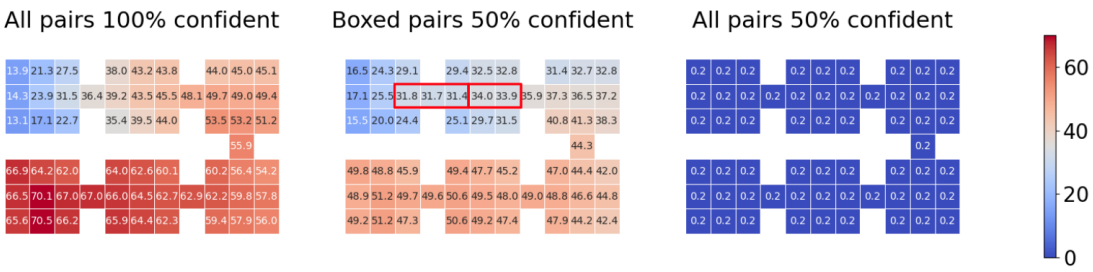


Figure 5: The heat maps showing that feedback inconsistency pushes rewards towards 0. Each grid in the map shows the score of the state where the agent is at this grid. The first heat map shows state scores trained with 100% confident rankings on all state pairs. The second heat map shows state scores trained with 100% confident ranking on all state pairs except 50% confident rankings on state pairs where the agent is in the red block. The third heat map shows state scores trained with 50% rankings on all state pairs.

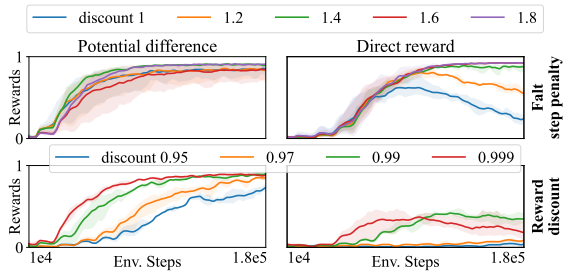


Figure 6: The average learning curves for rewards with multiple step penalties or discounts in the Grid World - Lock scenario, over 3 random seeds.

processing, a step penalty is essential; however, choosing this value can be difficult. We conduct further experiments in the grid world Lock scenario to show that our method is less sensitive to step penalty than direct reward. Two penalty schemes with multiple parameters are tested: 1) **Flat Step Penalty**: A positive constant is subtracted at each time step. 2) **Reward Discount**: Reward for episode step t is discounted by γ_r^t, where $\gamma_r < 1$ is a positive constant. We use the state score model trained from ground truth human heuristics for comparison. Each parameter is tested to train 3 RL policies with random seeds and initialization. The results are shown in Fig. 6.

Our method shows robustness to the choice of the flat step penalty, as the curves of penalty variances are less divergent. However, when using it as a direct reward, it can be seen that the performance is affected significantly with respect to the penalty, as many of them prevent the agent from converging. The results also show that our method can perform well by picking a commonly-used discount factor such as 0.99, avoiding the burden of extensive hyperparameter tuning. However, using it as a direct reward requires further hyperparameter tuning.

5.4 Multi-Query Evaluation

We introduce a multi-query approach that queries about ranking each state-transition case in the scoring-model training dataset a given number of times to address the rankings' inconsistency with lower-performing LLMs to push potential difference rewards toward zeros in the face of conflicting responses. In Fig. 5, we illustrate the heat maps of state scores trained with datasets of distinct consistency degrees, demonstrated in the Grid World MultiLock environment. Each grid in the heat maps records the score the scoring model assigns for an agent at that location. The left sub-figure demonstrates the ideal case in which 100% correct rankings are utilized to train the scoring model, demonstrating a smooth gradient from the start room (top left corner) to the final room (bottom left corner) roughly following the correct path. However, if the scoring model is trained with 50% confidence on all state pairs (right sub-figure in Fig. 5), the score in any state becomes equal as no adjacent states are ranked higher with high confidence. This demonstrates our method's ability to disregard states, and thus not provide rewards when LLM rankings are inconsistent across multiple queries. Finally, when the ranking results for a subset of states are inconsistent, yet consistent for all other locations (see Fig. 5 center), the correct gradual change in score is maintained outside of the affected area. These results underline our method's capabilities with respect to the effects of pushing uncertain state scores toward zero while giving contrasting rewards to confident pairs, ultimately improving performance of our method with low-performing LLMs (see Sec. 5.4.1).

5.4.1 Synthetic Ranking Evaluation

To test what ranking accuracy of datasets is needed for the LLM with multi-query methods, and how

Figure 7: The average learning curves with rewards trained from synthetic multi-query ranking datasets in the Grid World - MultiLock scenario, over 3 random seeds.

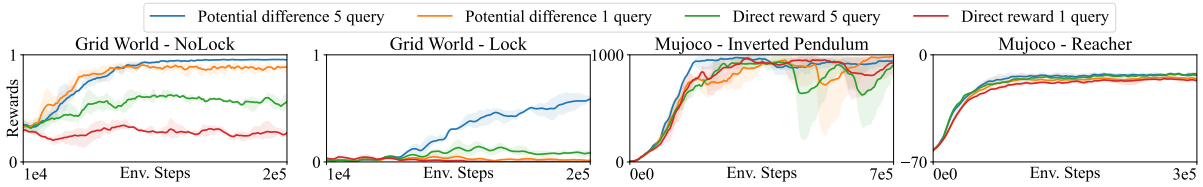


Figure 8: The average-performance comparison of 5-query variation of potential-difference rewards and direct rewards with 1800, 2200, 1000, 1000 state pairs ranked with Mixtral, over 5 random seeds.

many queries are required, we synthesized training datasets with specific ranking accuracy from 60% to 90% and simulated query times from 1 to 10. State-score models trained with these datasets output rewards when training RL policies, and their performance is shown in Fig. 7. The specific ranking correctness rates are controlled by introducing random ranking errors into the ground-truth ranking datasets. This approach is repeated on several copies of the ground-truth datasets to simulate the multi-query ranking results.

The result demonstrates that with more and more queries, the potential-difference reward gradually improves the training performance. Two or more queries may achieve fast policy training converging towards optimal if using ranking datasets with high feedback consistency to train state-score models. Notably, even for the datasets of only 60% ranking accuracy, which is close to random guessing, potential-difference rewards trained with enough queries can still increase the average policy training returns from 0 to an almost optimal level with 10 queries. This indicates that with enough queries, even the datasets with low-ranking accuracy can be boosted to function like those with high accuracy. This finding is consistent with our theoretic analysis and demonstrates considerable potential in mitigating significant ranking errors.

5.4.2 LLM Ranking Evaluation

Observing that Mixtral has the highest inconsistency in ranking states and thus has the largest potential for improvement, we evaluate the 5-query variations of potential-difference rewards and direct rewards with ranking results from Mixtral to verify our claims. Different methods' RL policy training curves averaged over 5 random seeds are compared in Fig. 8. As hypothesized, the 5-query potential-difference rewards achieve faster policy training and result in the highest rewards in all experiments. The single-query potential-difference rewards also outperform the single-query direct rewards. The improvement is most significant in Grid World - Lock scenario.

6 Conclusions

In this work, we propose a simple method for incorporating noisy LLM feedback into reinforcement learning. Our approach is based on learning a potential-based score function over repeated LLM-generated preference rankings which issues uninformative rewards for states in which the LLM is uncertain. We show both theoretically and empirically that this results in a natural trade-off between reward sparsity and LLM confidence in a variety of discrete and continuous environments.

7 Limitations

Our current analysis is limited to relatively simple discrete and continuous environments so that we could perform a thorough empirical evaluation. However, the consequence of this is that several of the LLMs, e.g., GPT-4, perform exceptionally well when ranking which leaves limited room for improvement. This is especially notable in the MuJoCo environments where our potential difference approach results in insignificant changes to performance. On the other hand, smaller-parameter models such as Mixtral exhibit worse performance and as such benefit more from our approach (Fig. 4) which is in-line with our synthetic experiments (Fig. 8). In the future, we wish to explore more complex, realistic environments which induce similar ranking errors in a larger set of language models. Our method is theoretically compatible with visual and multimodal environments that possess richer state and action spaces and local observations, which can be ranked by LLMs or VLMs. Exploring these scenarios will be the focus of our future work. A further limitation is that we currently assume that sequential state pairs can be randomly sampled from the environment. While this is true in most simulated environments, this assumption is violated in others such as the real world. In future work, we will explore iterative algorithms which alternate training the policy and sampling state pairs for ranking.

Acknowledgments

This work is sponsored by Honda Research 58629.1.1012949, AFOSR FA9550-18-1-0251 and DARPA FA8750-23-2-1015. The authors would also like to thank Dr. Woojun Kim for his assistance during discussion of this research.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. 2022. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

Erdem Bıyık, Dylan P Losey, Malayandi Palan, Nicholas C Landolfi, Gleb Shevchuk, and Dorsa Sadigh. 2022. Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *The International Journal of Robotics Research*, 41(1):45–67.

Erdem Bıyık, Malayandi Palan, Nicholas C Landolfi, Dylan P Losey, and Dorsa Sadigh. 2019. Asking easy questions: A user-friendly approach to active reward learning. *arXiv preprint arXiv:1910.04365*.

Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *Preprint*, arXiv:1606.01540.

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

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. 2023. Guiding pretraining in reinforcement learning with large language models. In *International Conference on Machine Learning*, pages 8657–8677. PMLR.

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

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Sungdong Kim, Sanghwan Bae, Jamin Shin, Soyoung Kang, Donghyun Kwak, Kang Min Yoo, and Minjoon Seo. 2023. Aligning large language models through synthetic feedback. *arXiv preprint arXiv:2305.13735*.

Martin Klissarov, Pierluca D'Oro, Shagun Sodhani, Roberta Raileanu, Pierre-Luc Bacon, Pascal Vincent, Amy Zhang, and Mikael Henaff. 2023. Motif: Intrinsic motivation from artificial intelligence feedback. *arXiv preprint arXiv:2310.00166*.

Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. 2023. Reward design with language models. *arXiv preprint arXiv:2303.00001*.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. 2023. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*.

Kimin Lee, Laura Smith, and Pieter Abbeel. 2021a. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. *arXiv preprint arXiv:2106.05091*.

Kimin Lee, Laura Smith, Anca Dragan, and Pieter Abbeel. 2021b. B-pref: Benchmarking preference-based reinforcement learning. *arXiv preprint arXiv:2111.03026*.

Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. 2018. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*.

Qing Lyu, Kumar Shridhar, Chaitanya Malaviya, Li Zhang, Yanai Elazar, Niket Tandon, Marianna Apidianaki, Mrinmaya Sachan, and Chris Callison-Burch. 2024. Calibrating large language models with sample consistency. *arXiv preprint arXiv:2402.13904*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Dorsa Sadigh, Anca D Dragan, Shankar Sastry, and Sanjit A Seshia. 2017. *Active preference-based learning of reward functions*.

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

Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. 2022. Defining and characterizing reward gaming. *Advances in Neural Information Processing Systems*, 35:9460–9471.

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

Gokul Swamy, Christoph Dann, Rahul Kidambi, Zhiwei Steven Wu, and Alekh Agarwal. 2024. A minimaximalist approach to reinforcement learning from human feedback. *arXiv preprint arXiv:2401.04056*.

Sree Harsha Tanneru, Chirag Agarwal, and Himabindu Lakkaraju. 2024. Quantifying uncertainty in natural language explanations of large language models. In *International Conference on Artificial Intelligence and Statistics*, pages 1072–1080. PMLR.

Andrea Lockerd Thomaz, Cynthia Breazeal, et al. 2006. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *Aaai*, volume 6, pages 1000–1005. Boston, MA.

Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Yufei Wang, Zhanyi Sun, Jesse Zhang, Zhou Xian, Erdem Biyik, David Held, and Zackory Erickson. 2024. Rl-vlm-f: Reinforcement learning from vision language foundation model feedback. *arXiv preprint arXiv:2402.03681*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. 2023. Text2reward: Automated dense reward function generation for reinforcement learning. *arXiv preprint arXiv:2309.11489*.

Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, et al. 2023. Language to rewards for robotic skill synthesis. *arXiv preprint arXiv:2306.08647*.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023. Siren's song in the ai ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*.

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

A Theoretical Proof: Inconsistent rankings lead to uninformative rewards

Lemma 1. *In the scope of RL based on LLM feedback, the confidence-based preference loss is equivalent to the standard preference loss used by state-score model training over multi-query ranking datasets.*

Proof. Given that the the confidence of ranking s_a higher than s_b, $conf\{y = (s_a \succ s_b)\}$, is defined as $\frac{N(s_a \succ s_b)}{N_{query}(s_a, s_b)}$. where $N(s_a \succ s_b)$ denotes the number of times LLM ranks s_a higher than s_b, and $N_{query}(s_a, s_b)$ denotes the total number of queries on s_a and s_b.

The standard preference loss over multiple-query ranking dataset $\mathcal{D}$ can be written as

$$\begin{aligned}
\mathcal{L}_{\mathcal{D}} = -\mathbb{E}_{(s_a,s_b,y)\sim\mathcal{D}}\Bigg[\mathbb{E}_{N_{query}}\Big[\mathbb{I}\{y = (s_a \succ s_b)\} \\
\log(sigmoid(\sigma_\psi(s_a) - \sigma_\psi(s_b)))+ \\
\mathbb{I}\{y = (s_b \succ s_a)\} \\
\log(sigmoid(\sigma_\psi(s_b) - \sigma_\psi(s_a)))\Big]\Bigg] \\
= -\mathbb{E}_{(s_a,s_b,y)\sim\mathcal{D}}\Bigg[\\
conf\{y = (s_a \succ s_b)\} \\
\log(sigmoid(\sigma_\psi(s_a) - \sigma_\psi(s_b)))+ \\
conf\{y = (s_b \succ s_a)\} \\
\log(sigmoid(\sigma_\psi(s_b) - \sigma_\psi(s_a)))\Bigg].
\end{aligned} \tag{5}$$

□

Theorem 1. *As inconsistency of a ranking over two states increases, the scores of these two states converge to the same value.*

Proof. Based on Equation 5,

$$\begin{aligned}
\mathcal{L}_{\mathcal{D}} = -\mathbb{E}_{(s_a,s_b,y)\sim\mathcal{D}}\Bigg[conf\{y = (s_a \succ s_b)\} \\
\log(sigmoid(\sigma_\psi(s_a) - \sigma_\psi(s_b)))+ \\
(1 - conf\{y = (s_a \succ s_b)\}) \\
\log(1 - sigmoid(\sigma_\psi(s_a) - \sigma_\psi(s_b)))\Bigg].
\end{aligned} \tag{6}$$

Take an arbitrary state pair (s_0, s_1) from $\mathcal{D}$. As inconsistency of the ranking over s_0 and s_1 increases, $conf\{y = (s_0 \succ s_1)\} \to 0.5$. Denote $sigmoid(\sigma_\psi(s_0) - \sigma_\psi(s_1))$ as $p_{0,1}$, $\mathcal{L}$ over other states in $\mathcal{D}$ without s_0, s_1 as $\mathcal{L}_{\mathcal{D}\setminus\{s_0,s_1\}}$,

$$\begin{aligned}
&\lim_{conf\{y=(s_0\succ s_1)\}\to 0.5} \mathcal{L}_{\mathcal{D}} \\
&\to -\frac{1}{2|\mathcal{D}|}\log(p_{0,1}(1-p_{0,1})) + \mathcal{L}_{\mathcal{D}\setminus\{s_0,s_1\}} \\
&\geq \frac{1}{|\mathcal{D}|}\log 2 + \mathcal{L}_{\mathcal{D}\setminus\{s_0,s_1\}}.
\end{aligned} \tag{7}$$

If and only if $p_{0,1} \to \frac{1}{2}$, where $\sigma_\psi(s_0) - \sigma_\psi(s_1) \to 0$, $\lim_{conf\{y=(s_0\succ s_1)\}\to 0.5} \mathcal{L}_{\mathcal{D}} \to \frac{1}{|\mathcal{D}|}\log 2 + \mathcal{L}_{\mathcal{D}\setminus\{s_0,s_1\}}$, reaching the lower bound.

Therefore, when training the state-score model with this loss $\mathcal{L}$, the scores of any two states whose ranking confidence is close to 50% will be pushed to the same value. □

B Scoring-Model Training Datasets

To train the scoring model, we randomly sample sequential state pairs and collect LLM ranking results on them, assembling all the data into a training dataset. The training data for all six environments can be accessed here: Scoring-Model Training Data. The details of collection process are as follows.

B.1 LLM Preference Generating Process

The LLM does pairwise state ranking in this work. We follow the methodology described in (Lee et al., 2023), where the LLM prompt consists of four parts:

1. **Preamble:** A description of the environment, task, and ranking criteria.
2. **Few-shot exemplar:** Pairwise state-ranking example, showcasing the chain of thought on ranking according to given environment conditions and state evaluation criteria.
3. **Sample to annotate:** The pair of specific states a and b, described with natural language.
4. **Ending:** Ending text to prompt a preferred response as ranking.

In the generated response, the LLM determines the ranking based on the specified criteria between two sequential states and outputs either 'Yes' (a is ranked higher than b) or 'No' (b is ranked higher

than a). The following is an example prompt for the Inverted Pendulum environment:

Preamble

You are in an environment, which involves a cart that can move linearly, with a pole fixed on it at one end and having another end free. The cart can be pushed left or right, and the goal is to balance the pole on the top of the cart by applying forces on the cart.

Few-shot exemplar

Q:
State[a]:
The pole leans to the left by 0.1 radians with a leftward velocity.

State[b]:
The pole stands vertically with a rightward velocity.

Is the transition from State[a] to State[b] a good transition?

A:
Yes, the pole currently stands vertically, so it has been balanced. Therefore, the answer is yes.

Sample to annotate

Q:
State[0]:
The pole leans to the right by 0.6 radians with a leftward velocity.

State[1]:
The pole leans to the right by 0.1 radians with a rightward velocity.

Ending

Is the transition from State[0] to State[1] a good transition?

Prompts for other environments can be accessed here: Prompts for all 6 environments. To maintain consistent settings across all experiments and eliminate the influence of irrelevant variables, we use the same prompts for all LLM models.

C Experiment Details for Reproducibility

C.1 Model Architecture

Grid World The policy model for all scenarios contains separate actor and critic networks, both with 3 convolutional layers followed by 1 fully connected layer mapping the flattened vector to the output vector. The convolutional layers consist of 16 2×2 filters, followed by 2×2 pooling, then 32 2×2 filters, and finally 64 2×2 filters. Scoring model architectures for each scenario are shown in Table 2.

	NoLock Lock	MultiLock
Conv	conv 16, (2,2), pool (2,2), conv 32, (2,2), conv 64, (2,2), conv 128, (2,2)	conv 16, (2,2), pool (2,2), conv 32, (2,2), conv 64, (2,2)
FC hidden	256, 128, 64, 32, 16	512, 256, 128, 64, 16

Table 2: Scoring model architecture for Grid World scenarios. The number of output channels and kernal size is given for each convolutional layer. The number of nodes for each fully connected hidden layer are given.

MuJoCo Policy model follows Schulman et al. (2017), where both actor and critic networks have a fully connected network using a hidden layer with 64 nodes. Distinct scoring model architectures are used in each scenario, shown in Table 3.

	Reacher	Inverted Pendulum	Hopper
FC hidden	128	128	128, 64

Table 3: Scoring model architecture for MuJoCo scenarios. The number of nodes for each fully connected hidden layer are given.

C.2 Hyperparameters

The hyperparameters of training scoring models and PPO policies were tuned manually. The details are recorded in Table 4, 5, 6, 7.

Hyperparameter	NoLock	Lock	MultiLock
Learning Rate	[0.0028, 150000] [0.00000001, 180000]	[0.008, 100000] [0.001, 127000], [0.000005, 180000]	0.0001
Batch Size	1024	2048	2048
Num. SGD Epochs	4	4	4
Minibatch Size	128	128	128
Clipping Prameter	0.2	0.2	0.2
VF Clip Parameter	10.0	10.0	10.0
VF Coeff.	0.5	0.5	0.5
KL Coeff.	0.5	0.5	0.5
Entropy Coeff.	0.01	0.01	0.01
GAE	0.8	0.8	0.8
Discount	0.99	0.99	0.99

Table 4: PPO hyperparameters for Grid World scenarios.

Hyperparameter	NoLock	Lock	MultiLock
SM. LR. Schedule	[0.004, 17] [0.0001, 45], [0.0000001, 250]	[0.000000001, 20] [0.00004, 45], [0.0000001, 250]	[0.000004, 20], [0.000004, 100], [0.0000001, 250]
SM. Batch Size	32	16	64
SM. Epochs	200	120	250

Table 5: Scoring model training hyperparameters for Grid World scenarios. Learning rate schedule is presented as the learning rate value along with the corresponding final epoch it is applied to.

Hyperparameter	Value
Learning Rate	0.0003
Batch Size	2048
Num. SGD Epochs	10
Minibatch Size	64
Clipping Prameter	0.2
VF Clip Parameter	10
VF Coeff.	1
KL Coeff.	0.2
Entropy Coeff.	0
GAE	0.95
Discount	0.99

Table 6: PPO hyperparameters for all 3 MuJoCo environments

Hyperparameter	Pendulum	Reacher	Hopper
SM. LR. Schedule	[0.000004, 20] [0.00002, 40], [0.000008, 50]	[0.000004, 20] [0.00002, 120], [0.000008, 300]	[0.000004, 50], [0.00002, 70]
SM. Batch Size	16	16	16
SM. Epochs	50	300	70

Table 7: Scoring model training hyperparameters for Mujoco scenarios. Learning rate schedule is presented as the learning rate value along with the corresponding final epoch it is applied to.