Specifying Reward Functions for RL Without Environment Sampling

Stephane Hatgis-Kessell, W. Bradley Knox, Emma Brunskill
stephhk@stanford.edu

Abstract

Enabling human stakeholders to specify reward functions that lead to their desired outcomes is a key challenge in deploying reinforcement learning agents. Preference-based methods such as online RLHF can reduce the burden of manual reward design, but they require repeatedly training policies, sampling trajectories from the real world, and eliciting feedback, making them impractical in settings where environment interaction is computationally expensive or unsafe. We introduce Experience-Free Automonous Reward Specification (EARS), a method for learning reward functions from preferences without environment interaction. Our approach uses a structured LLM-mediated process to construct a small set of expressive reward features from a task description and the environment observation space, then strategically samples imagined trajectories in this feature space and learns feature weights from preferences over the imagined trajectory pairs. We evaluate on three long-horizon domains: pandemic lockdown regulation design, insulin administration for diabetes patients, and autonomous vehicle control on a highway. We compare EARS to baselines that also enable reward specification without environment interaction—namely, methods that directly prompt an LLM to generate a reward function. When learning from either ground-truth preference labels or preferences labeled by a large language model (LLM), EARS designs reward functions that are more aligned with the ground truth reward function that produced the preferences or LLM context than these baselines. These results suggest that preference-based reward specification remains effective without environment sampling, enabling practical reward design in settings where collecting real trajectories is costly or infeasible.

1 Introduction

A central challenge in AI is enabling human stakeholders to specify their objectives (Amodei et al., 2016; Leike et al., 2018). A common approach is to encode these objectives as a reward function and optimize it via reinforcement learning (RL). Unfortunately, hand-designing a reward function is difficult, and can often result in a decision policy that does not behave as intended. This has motivated extensive interest in learning a reward function from preferences such as in RLHF (Christiano et al., 2017; Lee et al., 2021; Ibarz et al., 2018; Pacchiano et al., 2021; Dong et al., 2024; Sadigh et al., 2017). In particular, in *online RLHF*, a reward model is learned from human preference data and used to train a policy. The resulting policy is then rolled out in a real environment to generate trajectories. For example, the policy might control a humanoid learning to cook food, or generate a disaster response for extreme weather events modeled in a complex physics-based simulator. The resulting trajectories (of cooking, or disaster response in an expensive simulator) can then be used to elicit additional human feedback. The reward model is updated with this new data, and the process iterates.

However, in many domains this process involves considerable cost. For example, for humanoid robotics environments or those that use expensive physics simulators, policy rollouts are expensive or risky. This can make both policy training given a fixed reward function and data collection of

trajectories to elicit preferences over extremely costly. Therefore, when training a reward model requires many rounds of data collection, online RLHF can be difficult to apply in practice..

These challenges may also directly translate into implications for data-efficiency; long policy training times may require eliciting larger batches of preferences to limit the number of policy updates, which in turn may require more human preference labels to learn desirable behavior (Citovsky et al., 2021; Biyik et al., 2024). Further, constructing a dataset for preference elicitation requires sampling sufficiently diverse trajectories to elicit feedback over, including sub-optimal data. For many real-world domains, collecting large amounts of sub-optimal data is prohibitive due to safety concerns (Garcia & Fernández, 2012). If a high-fidelity simulator or world model is not available for data collection—which often require collecting large amounts of real world data to build—then acquiring the diverse and sufficiently sub-optimal trajectories required for effective preference elicitation may be very expensive (Cosner et al., 2022; Liu et al., 2023). Offline-RLHF suffers from the same limitations, i.e., it requires sampling large amounts of sufficiently diverse experience.

In this paper, we consider how to learn aligned reward functions without environment interaction—relevant when generating policy rollouts in the real domain is expensive. We introduce Experience-free Autonomous Reward Specification (EARS), which still assumes access to preference labels but learns a reward function without sampling from the environment. EARS comprises two novel stages. The first stage is a structured interaction between Large Language Models (LLMs) to construct a set of reward features that can represent different reward functions. Then in the second stage we strategically sample imagined trajectories from this generated feature space to elicit preferences over and construct a reward function.

We evaluate EARS in three complex, long horizon decision-making tasks with preferences labeled either synthetically by a ground-truth reward function, or by an LLM conditioned on a natural language specification of the task objectives—the kind of document a careful human stakeholder could plausibly produce. We first show that prompting an LLM to generate a reward function—even with an exceedingly specific description of the ground-truth reward—often fails to produce a well aligned reward function, highlighting the limitations of the only alternative available when environment interaction is unavailable. We then show that EARS produces reward functions far more closely aligned with the ground truth when learning from ground-truth preferences or LLM labeled preferences. Moreover, EARS matches or outperforms methods that do sample real trajectories from the environment to elicit preferences over. Our contribution is three-fold:

- We show that directly prompting an LLM to design a reward function with an extremely detailed natural language description often does not produce a more aligned reward function than providing no description.
- We introduce EARS, to design reward functions with preferences without sampling from an environment. EARS samples zero environment transitions, making it strictly more environment-sample-efficient than methods like online RLHF. We show that when learning from ground-truth preferences it matches the preference-label efficiency of a baseline that learns from real trajectories and outperforms directly prompting an LLM.
- We demonstrate that, when only given access to a natural language specification of the task objectives, EARS designs more aligned reward functions than directly prompting an LLM with the same specification

We will release all code to reproduce experiments upon release of the paper.

2 Preliminaries

Consider an MDP $\mathcal{M} \triangleq (S, A, \Omega, \gamma, p_0, r)$ with state space S, action space A, transition dynamics $\Omega : S \times A \to \Delta(S)$, discount factor $\gamma \in [0, 1]$, and initial state distribution p_0. Let τ denote a trajectory $\tau = (s_0^\tau, a_0^\tau, s_1^\tau, a_1^\tau, \ldots)$ starting at $s_0^\tau \sim p_0$. We allow τ to be finite (e.g., when terminating in an absorbing state) or infinite. In this work, we consider the undiscounted setting

($\gamma = 1$); the formulation extends naturally to the discounted case ($\gamma < 1$). The ground-truth reward function is $r : S \times A \times S \to \mathbb{R}$. We denote by $\mathcal{M} \setminus r \triangleq (S, A, \Omega, \gamma, p_0, _)$ the environment without a specified reward function. Let r be the (unobservable) ground-truth reward function, $\hat{r}$ a learned approximation, and $\tilde{r}$ an arbitrary reward function. We denote the reward function produced by EARS as a linear functions over features: $\hat{r}(s, a, s') = \hat{w}^\top \phi(s, a, s')$ where $\phi(s, a, s') \in \mathbb{R}^d$ is a feature representation and $\hat{w} \in \mathbb{R}^d$ is a weight vector. The ground-truth reward functions and the reward function produced by the other methods we consider may be non-linear. We do not assume access to a known feature representation that can be used to construct the ground-truth reward function; this work concerns both learning $\hat{w}$ and specifying ϕ. A policy $\pi : S \times A \to [0, 1]$ maps states to action distributions. Its expected discounted return under $\tilde{r}$ from start distribution p_0 is $J_{\tilde{r}}(\pi)$. An *optimal policy* for $\tilde{r}$ is any $\pi^*_{\tilde{r}} \in \arg\max_\pi J_{\tilde{r}}(\pi)$.

As a complement to τ, let Φ denote a trajectory expressed as the sum of discounted reward features. For example, for a given trajectory, $\Phi_\tau = \sum_{t=0}^{|\tau|-1} \gamma^t \phi(s^\tau_t, a^\tau_t, s^\tau_{t+1})$ where $|\tau|$ denotes the number of transitions in τ (possibly infinite). Going forward, we will only consider trajectories expressed as the sum of reward features, rather than as a list of transitions. We learn a reward function $\hat{r}$ from trajectory pair preferences. Let $\mathcal{D} = \{(\Phi_1, \Phi_2, \mu)\}_{k=1}^N$, and $\mu \in \{0, 1, \frac{1}{2}\}$ with $0 : \Phi_1 \succ \Phi_2,\ 1 : \Phi_2 \succ \Phi_1,\ \frac{1}{2} : \Phi_1 \sim \Phi_2$. where $\succ$ means is "preferred to" and $\sim$ means is "equally preferred to".

We consider two different approaches to learning a reward function from the preference dataset.

Learning from the Bradley–Terry preferences A standard assumption in RLHF is that preferences are generated by the Bradley–Terry preference model (Christiano et al., 2017):

$$P(\Phi_1 \succ \Phi_2 \mid \tilde{w}) = \sigma\big(\tilde{w}^T \Phi_1 - \tilde{w}^T \Phi_2\big), \quad \sigma(x) = 1/(1 + e^{-x}) \tag{1}$$

and that $\hat{w}$ is learned by minimizing the cross-entropy loss:

$$\mathcal{L}_{\text{pref}}(\hat{w}; \mathcal{D}_t) = -\sum_{(\Phi_1, \Phi_2, \mu) \in \mathcal{D}} \mu \log P(\Phi_1 \succ \Phi_2 | \hat{w}) + (1 - \mu) \log P(\Phi_1 \prec \Phi_2 | \hat{w}). \tag{2}$$

While widely used, the assumption that preferences are generated according to the Bradley–Terry model can be limiting; see Zhi-Xuan et al. (2024) for a detailed critique. Nevertheless, when evaluating reward specification from ground-truth preferences—i.e., preferences generated by a ground-truth reward function to approximate those a real human might provide—we adopt the Bradley–Terry model and learn a reward function by minimizing the cross-entropy loss.

Learning from noiseless preferences Assuming preferences are sampled from the Bradley-Terry preference model relies on the assumption that preferences follow a particular type of structured noise. Here, we outline a method for learning a reward function from preferences that are noiselessly generated by a reward function, as has been assumed by some prior work (Knox et al.; Hatgis-Kessell et al.). In particular, we assume:

$$P(\Phi_1 \succ \Phi_2 \mid \tilde{w}) = \mathbb{1}\big\{\tilde{w}^T \Phi_1 > \tilde{w}^T \Phi_2\big\}, \tag{3}$$

When preferences are labeled by an LLM, rather than by humans or synthetic annotators, we posit that this constitutes a strong alternative to the Bradley–Terry assumptions, which were originally developed to model human behavior—albeit are still flawed (Zhi-Xuan et al., 2024). Following the observation of Kim et al. (2024), given preference dataset $\mathcal{D}$, we can learn a reward function by solving the following linear program:

$$\begin{aligned}
\text{find} \quad & \hat{w} \in \mathbb{R}^d \\
\text{s.t.} \quad & \hat{w}^\top (\Phi_{1,k} - \Phi_{2,k}) \geq \epsilon, \quad \forall k \text{ where } \mu_k = 0, \\
& \hat{w}^\top (\Phi_{1,k} - \Phi_{2,k}) \leq \epsilon, \quad \forall k \text{ where } \mu_k = 1, \\
& \hat{w}^\top (\Phi_{1,k} - \Phi_{2,k}) = 0, \quad \forall k \text{ where } \mu_k = \tfrac{1}{2}, \\
& \|\hat{w}\|_1 \leq B.
\end{aligned} \tag{4}$$

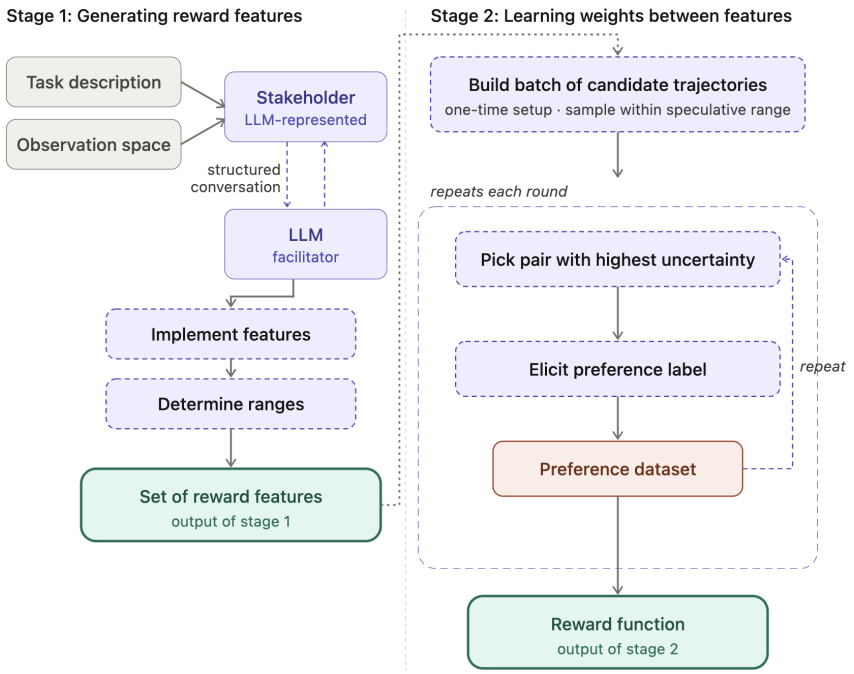

Figure 1: **Experience-free Autonomous Reward Specification (EARS)**: In Stage 1, a natural language task description and a description of the environment observation space are inputted to an LLM-represented stakeholder. The LLM-represented stakeholder deliberates with a LLM-represented facilitator to design a small set of reward features that can be combined linearly to represent different objectives. In Stage 2, imagined trajectory pairs are sampled from the designed feature space and preferences are labeled over those pairs. The output of Stage 2 is a reward function learned from those preferences.

where ϵ is a tolerance parameter.

3 Experience Free Automatic Reward Specification (EARS)

EARS consists of two stages, illustrated in Figure 1. In the first stage, we engage an LLM, representing a human stakeholder or group of stakeholders, in a structured conversation with another LLM, representing a facilitator. The aim of the conversation is to design a small set of expressive and interpretable features that, when combined linearly, can represent the different objectives a stakeholder may have. In the second stage we elicit preferences to learn the weights between those reward features. The output of the second stage is a reward function learned to satisfy the elicited preferences. The environment is never sampled to produce the reward function.

3.1 Stage 1: Designing a set of reward features

The input to stage 1 is a brief description of the task, a description of the environment observation space represented as Python code, and optionally, a description of the task objective, supplied by a human stakeholder. The output of stage 1 is a small set of interpretable reward features: $\phi(s_t, a_t, s_{t+1}) = \{\phi_1(s_t, a_t, s_{t+1}), ..., \phi_d(s_t, a_t, s_{t+1})\}$. Each reward feature ϕ_i is implemented as a python function, and is later linearly combined to produce a reward function.

To generate the set of reward features, we engage a stakeholder and facilitator, both represented as LLMs, in a structured conversation. The conversation seeks to: (1) unearth the different objectives people may have, (2) determine which objectives are measurable given the environment's observation space, and (3) aggregate and implement the identified objectives as executable Python functions. Additionally, to enable the sampling procedure used in stage 2, stage 1 also outputs a

speculative minimum and maximum range of the sum of each reward feature over of a trajectory of H steps—denoted as $\phi^H_{\text{min}} \in \mathbb{R}^d$ and $\phi^H_{\text{max}} \in \mathbb{R}^d$. We prompt the stakeholder to select finite values that represent the likely ranges that would be observed.

3.2 Stage 2: Learning weights over the reward features

The input to stage 2 is the set of generated reward features from stage 1 and a preference labeler—whether that be a human stakeholder, an LLM, or something else. The output of stage 2 is a set of weights over the set of generated reward features, i.e., a reward function.

Constructing imagined trajectories Our key insight is that, given a small set of generated reward features, we can construct "imagined" trajectories by sampling feature values uniformly within their feasible ranges. Specifically, for each feature, we uniformly sample values from a speculated minimum–maximum range determined in Stage 1. We denote an imagined trajectory as $\Phi \in \mathbb{R}^d$, where d is the number of reward features, which is constructed by sampling from the uniform distribution $\mathcal{U}$ over reward feature ranges: $\Phi \sim \mathcal{U}([\phi^H_{\text{min}}, \phi^H_{\text{max}}])$. While many of these imagined trajectories may not be physically plausible, preferences over them can still be useful for learning a reward function.

Strategically sampling imagined trajectories To avoid requiring prohibitively large datasets of preferences, prior work in online RLHF focuses on strategically sampling trajectory pairs that can reduce uncertainty (Christiano et al., 2017; Lee et al., 2021; Ibarz et al., 2018; Pacchiano et al., 2021; Dong et al., 2024; Sadigh et al., 2017). This approach is particularly apt for imagined trajectories, since sampling them is cheap and we can draw many candidates and select the single most informative pair. At each round of feedback collection, we sample k imagined trajectories and then select the pair with the highest uncertainty under the current dataset $\mathcal{D}$, according to an uncertainty function $f(\Phi_1, \Phi_2, \mathcal{D})$: $(\Phi_1^*, \Phi_2^*) = \arg\max_{\Phi_1,\Phi_2} f(\Phi_1, \Phi_2, \mathcal{D})$ where $f(\Phi_1, \Phi_2, \mathcal{D})$ is the subsequent sections. We restrict the selection to pairs of imagined trajectories that differ in at most two features; we speculate that this makes the salient differences between any two imagined trajectories easier for a preference labeler to reason about. A preference label is elicited over the selected imagined trajectory pair, added to the preference dataset, and then the process is repeated. $f(\Phi_1, \Phi_2, \mathcal{D})$ can be instantiated using any uncertainty measure proposed in prior work on online-RLHF. In this work, we consider two such measures, depending on whether preferences are assumed to be stochastic or noiseless.

Strategically sampling imagined trajectories with stochastic preferences When assuming preferences follow the Bradley-Terry preference model, we quantify uncertainty as the variance in predicted preference probabilities across an ensemble of reward models. In this work, the ensemble is maintained using an Epistemic Neural Network (ENN) (Osband et al., 2023) which provides a computationally efficient method for measuring epistemic uncertainty as the variance across different model predictions. Concretely, let $\{\hat{r}_{\theta^{(i)}}\}_{i=1}^M$ denote the ensemble of reward models, and define

$$p^{(i)}(\Phi_1 \succ \Phi_2) = \frac{\exp\left(\hat{r}_{\theta^{(i)}}(\Phi_1)\right)}{\exp\left(\hat{r}_{\theta^{(i)}}(\Phi_1)\right) + \exp\left(\hat{r}_{\theta^{(i)}}(\Phi_2)\right)}.$$

Then, $f_{\text{enn}}(\Phi_1, \Phi_2, \mathcal{D}) = \text{Var}_{i\in[M]}\left[p^{(i)}(\Phi_1 \succ \Phi_2)\right]$. This variance-based uncertainty measure is common in prior work (Christiano et al., 2017; Lee et al., 2021; Dwaracherla et al., 2024).

Strategically Sampling Imagined Trajectories with Noiseless Preferences Alternatively, when assuming preferences are noiseless and learned by solving a linear program via Eq. 4, we quantify uncertainty using the disagreement among the reward functions that are still consistent with $\mathcal{D}$. Each candidate reward function w assigns the pair a signed return difference $w^\top(\Phi_1 - \Phi_2)$, whose sign encodes which trajectory w prefers. Let $\mathcal{W}(\mathcal{D})$ denote the feasible set of weights satisfying the constraints from $\mathcal{D}$, i.e., the solutions to Eq. 4, and let ℓ and u be the smallest and largest such differences over this set:

$$\ell = \min_{w\in\mathcal{W}(\mathcal{D})} w^\top(\Phi_1 - \Phi_2), \quad u = \max_{w\in\mathcal{W}(\mathcal{D})} w^\top(\Phi_1 - \Phi_2).$$

We then define

$$f_{\mathrm{LP}}(\Phi_1, \Phi_2, \mathcal{D}) = \begin{cases} u - \ell & \text{if } \mathrm{sign}(\ell) \neq \mathrm{sign}(u), \\ 0 & \text{otherwise.} \end{cases}$$

A differing sign means some feasible reward function prefers Φ_1 while another prefers Φ_2, so the pair is contested and f_{LP} measures the magnitude of that disagreement. When $\mathrm{sign}(\ell) = \mathrm{sign}(u)$ (e.g., $\ell = 1$, $u = 10000$), every feasible reward function assigns the difference the same sign and therefore agrees on the ordering; the pair is uninformative and we set its uncertainty to zero.[1]

Learning a reward function from preferences over imagined trajectories The resulting dataset $\mathcal{D}$ consists of tuples $(\Phi_1, \Phi_2, \succ)$. A reward function is learned from this dataset using either the objective in Eq. 2 when assuming Bradley–Terry preferences, or the linear program in Eq. 4 when assuming noiseless preferences. Let $\hat{w}_{\mathcal{D}}$ be the weight vector that satisfies the preferences between imagined trajectory pairs in dataset $\mathcal{D}$ found with either reward learning method.[2] The transition-level reward function, which inputs transitions from the environment, is then $\hat{r}(s_t, a_t, s_{t+1}) = \hat{w}_{\mathcal{D}}^{\top}\phi(s_t, a_t, s_{t+1})$.

4 Experiments

We now empirically evaluate EARS. We consider two ways a stakeholder may specify a reward function: via preferences (Section 4.4)—which we study by simulating human preferences with a ground-truth reward function—and via a natural language specification of the task objectives, which serves as context for an LLM-based preference labeler (Section 4.5). This second setting represents a directly deployable use of EARS: the only inputs required are an environment description and a specification document a careful human could write, with no environment sampling and no human-in-the-loop preference labeling. All experiments use Gemini 3. We summarize our experiments and results in Table 5.

4.1 Evaluating the learned reward function.

We evaluate the alignment between a learned reward function $\hat{r}$ and the ground-truth reward function r using the Trajectory Alignment Coefficient (TAC) (Muslimani et al., 2025). TAC computes the Kendall-Tau correlation between the ranking of trajectories under the learned reward function,$\hat{r}$, and the ground truth reward function, r, where all trajectories share the same start state. We compute TAC using 500 trajectories sampled from the environment, denoted $\mathcal{T}_{\text{eval}}$. These trajectories are collected from checkpoints while training a policy with the ground-truth reward function, and are used solely for evaluation; consistent with our problem setting, no environment samples are used for learning the reward function. Our aim is to develop a method to achieve the following:

Objective. $\max_{\hat{r}} \ \mathrm{TAC}(\hat{r}, r)$ s.t. $\hat{r}$ is learned without sampling transitions from $\mathcal{M}$.

We further discuss this evaluation metric in relation to others in Section 6.

4.2 Environments

We evaluate EARS across three benchmark environments used in Pan et al. (2022), summarized in Table 1. For each environment, we consider two different ground-truth reward functions to align with, denoted as r_1 and r_2. These two ground-truth reward functions per environment are also designed by Pan et al. (2022) and are given in Section 9. All environments have large, continuous observation spaces that must be reasoned over to design a small set of reward features.

[1] As a fallback, if no trajectory pairs are found with non-zero uncertainty, we do not zero out the uncertainty.

[2] When solving the linear program in Eq. 4, $\hat{w}_{\mathcal{D}}$ need not be unique.

4.3 Baselines

Baselines that don't sample from environment To the best of our knowledge, the only existing class of methods that specifies a reward function without environment sampling is to directly prompt an LLM to write a reward function (e.g., Ma et al. (2023); Kwon et al. (2023); Xie et al. (2023); Yu et al. (2023)). These methods typically then refine the generated reward function through environment sampling—for example, by executing the proposed reward, observing the resulting behavior, and iterating. We introduce three baselines in this class but with the environment-sampling step removed to match our problem setting. Each baseline takes as input a task description and a description of the environment observation space, and outputs a reward function implemented in Python. Two of the baselines supply the reward-designing LLM with a natural language specification describing the task objectives and how they should be traded off—the same specifications that we evaluate EARs with in Section 4.5.

- **Direct Prompting:** Prompts an LLM to design a reward function for an RL agent, given only the task description and observation space. No stakeholder specification is provided.
- **Direct Prompting + Realistic Reward Description:** The reward-designing LLM is given a realistic stakeholder specification, produced by a separate LLM that fills out a structured template using a summary of the ground-truth reward. This specification represents what we believe a careful human stakeholder could plausibly produce in the realistic setting where no ground-truth reward function exists. The realistic reward descriptions are given in Appendix Figures 4, 8, and 12.
- **Direct Prompting + Privileged Reward Description:** The reward-designing LLM is given a highly detailed objective specification. To generate this specification, a separate LLM is shown the ground-truth reward implementation and asked to describe it in natural language. The resulting specification is likely more detailed than what a real human stakeholder without access to a ground-truth reward function could specify; it serves as an upper bound on the specificity available to a direct-prompting method. The privileged reward descriptions are given in Appendix Figures 3, 7, and 11.

In real-world use, the specification for the last two baselines would come directly from a human stakeholder, and no ground-truth reward function would exist.

Baselines that do sample from the environment We compare EARS to two baselines that, unlike EARS, learn from real trajectories sampled from the environment. Both baselines learn from preferences labeled by the ground-truth reward function over the same real trajectory pairs. The trajectories are sampled uniformly from a candidate pool built by rolling out policy checkpoints saved while training a policy with the ground-truth reward function, and pairs are uniformly sampled from this set. The two baselines differ only in the trajectory representation input to the model they learn; both learn a model parameterized as a non-linear neural network. Offline-RLHF learns a reward model defined over transitions, which can be used directly as a reward function. Learning To Rank Real Trajectories (LTRRT) instead represents each trajectory as the sum of its observation-space features, potentially avoiding challenges in correctly assigning trajectory return across transitions. ; because this model is defined over a full trajectory rather than transitions, it cannot itself be used as a reward function, and serves only to rank trajectories. [3] Note that both baselines enjoy an advantage unavailable to EARS: their preference pairs are drawn from real trajectories produced by a policy trained on the ground-truth reward function, whereas EARS—with no access to the ground-truth reward function—only constructs imagined trajectories. More details are provided in Appendix 8.

4.4 Results when specifying a reward function via ground-truth preferences

We evaluate EARS when assuming preferences are sampled from the Bradley-Terry preference model under the ground-truth reward function and when assuming preferences are noiselessly gen-

[3] We can still compare LTRRT to EARS because the TAC, our primary evaluation metric, only inputs an ordering over trajectories

Name	Objective	H	State	Action	r_1 Summary	r_2 Summary	Simulator
Pandemic Mitigation (Kompella et al., 2020)	Design COVID-19 pandemic lockdown regulations	192	Cont., 312-dim	Disc. $\{-1, 0, 1\}$	Trades off between lockdown regulation strictness and infection rates	Encourages strict and non-fluctuating lockdown regulations even when infection rates are low	Modified SEIR simulator (Kompella et al., 2020)
Glucose Monitoring (Man et al., 2014; Fox et al., 2020)	Administer insulin to patient with Type II diabetes	5760	Cont., 96-dim	1-d Cont. in $[0, 1]$	Prioritizes minimizing health risks	Prioritizes reducing financial cost of treatment and risk of death	FDA-approved simulator (Man et al., 2014; Fox et al., 2020)
Traffic Control (Wu et al., 2021)	Control autonomous vehicle (AV) fleet on highway	300	Cont., 50-dim	10-d Cont. in $[0, 1]$	Trades off between maximizing mean velocity of all vehicles and maintaining safe distances	Maximizes mean velocity	FLOW highway simulator (Wu et al., 2021)

Table 1: Environments, used by Pan et al. (2022). H = horizon.

erated by the ground-truth reward function. Because the ground-truth reward function inputs real transitions from the environment, but we elicit preferences over imagined trajectories, we learn a mapping between the two representations as detailed in Appendix Section 7.

Table 2: **EARS with preferences compared to prompting baselines:** TAC between learned and ground-truth reward functions across three environments, evaluated for two ground-truth reward functions (r_1 and r_2). We compare EARS against three prompting baselines: direct prompting with no reward description (No-RD), with a privileged reward description (Privileged-RD) and with a realistic reward description (Realistic-RD). We evaluate EARS with noiseless preferences and stochastic preferences. None of the methods sample transitions from the environment. Results are averaged over three random seeds, with each seed rerunning the full method including all LLM generations. Higher TAC indicates closer alignment with the ground-truth reward function.

		Pandemic		Glucose		Traffic	
Method	Variant	r_1	r_2	r_1	r_2	r_1	r_2
EARS	1k noiseless prefs.	.72±.01	**.96±.00**	.69±.07	.82±.04	**.92±.07**	.90±.09
	2k stochastic prefs.	**.73±.01**	.94±.02	**.74±.03**	.71±.06	.85±.04	.90±.09
Direct Prompting	No-RD	−.08±.12	.43±.12	.16±.20	−.20±.20	.25±.06	.67±.27
	Realistic-RD	−.37±.10	.68±.06	.27±.29	.79±.14	.27±.04	.94±.01
	Privileged-RD	−.14±.14	.58±.19	.21±.17	**.84±.12**	.17±.01	**1.00±.00**

Results are shown in Table 2, demonstrating that EARS learns more aligned reward functions from the ground-truth preferences than other methods that also do not rely on environment sampling. Notably, even providing the Direct Prompting baseline with an exceedingly specific objective description for r_1 (Privileged-RD) fails to generate an aligned reward function in all environments. EARS slightly under performs the direct prompting baselines when all methods produce reward functions with higher TAC, and significantly outperforms them otherwise.

Comparison to sampling trajectories from the real environment A natural concern is that EARS, by learning from imagined rather than real trajectories, might require many more preferences to learn the same reward function. We test this by comparing the data-efficiency of EARS against Offline-RLHF and Learning To Rank Real Trajectories, both of which learn from real environment trajectories, in Figure 2. We find that learning from real trajectories does not require substantially fewer preferences: EARS is in fact substantially more data-efficient than Offline-RLHF, and tends to match or outperform Learning To Rank Real Trajectories.

Comparison to sampling imagined trajectories from the environment observation space Stage 1 of EARS aims to produce a small set of reward features from which we can sample imagined trajectories in Stage 2. Here, we compare EARS to a method that uses the environment's observation

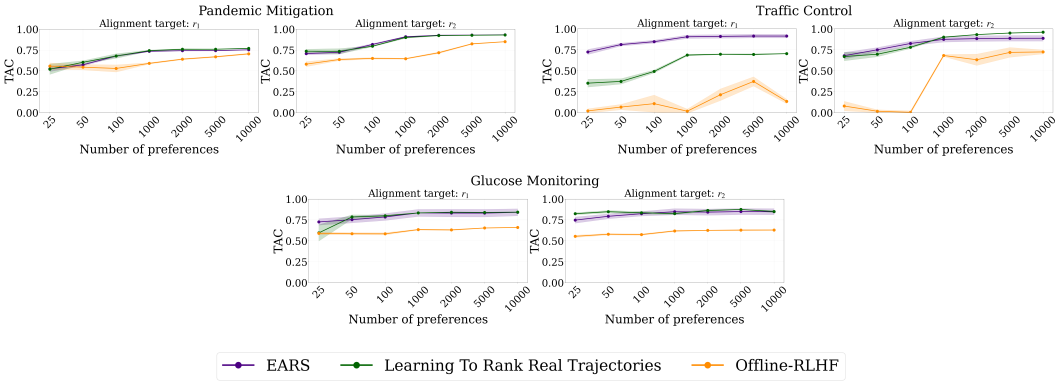

Figure 2: **Comparison to learning with real trajectories:** We compare EARS, which learns from preferences over imagined trajectories, against Offline-RLHF and Learning To Rank Real Trajectories, which both learn a non-linear model from trajectory pairs sampled from policy checkpoints in the environment. Results are shown for 5 seeds and the shaded area shows the standard error.

Table 3: **Effect/Ablation of designed reward features:** We compare EARS, which samples imagined trajectories from the LLM-designed feature space produced in Stage 1 (denoted as Designed feats), against an ablation that instead samples imagined trajectories directly from the environment's observation space variables, bypassing Stage 1 (denoted as Raw obs). See Table 2 for more details.

		Pandemic		Glucose	
Method	Variant	r_1	r_2	r_1	r_2
EARS, 2k stochastic prefs.	Designed feats	**.73±.01**	**.94±.02**	**.74±.03**	**.71±.06**
	Raw obs	−.42±.00	−.05±.02	.44±.01	.07±.04

space variables as features to sample from rather than executing Stage 1. Results are shown in Table 3, indicating that sampling imagined trajectories from the environment's observation space significantly under performs sampling imagined trajectories from the LLM designed feature space. The raw observation space is significantly larger, containing many redundant features unrelated to any objective. This makes informative sampling in Stage 2 harder: analogous to the noisy-TV problem in exploration, the uncertainty criterion is drawn toward pairs differing along these irrelevant features, where reward-function disagreement is high but uninformative about the ground-truth objective.

4.5 Results when specifying a reward function via natural language

We now evaluate EARS in a setting where it is directly deployable. Here we assume the input to EARS is a natural language specification of an objective, for evaluation purposes designed from the ground-truth reward function to emulate the specifications a human stakeholder could provide. The specification is input to an LLM that will label preferences instead of a ground-truth preference labeler (Section 4.4) or a human stakeholder. We evaluate EARS with the Privileged Reward Description and Realistic Reward Description outlined in Section 4.3, matching the corresponding baselines. All other evaluation details match those in Section 4.4.

Table 4 shows the results. EARS produces more aligned reward functions than the direct prompting baselines in nearly all settings, and yields substantially better performance when alignment with the ground-truth reward function is otherwise low. The exception is the traffic environment with the realistic reward description for r_1, where EARS underperforms direct prompting. Overall, EARS generally outperforms direct prompting, though at the cost of substantially more LLM generations for preference labeling.

Question & Finding	Results
Main comparisons	
Given ground-truth preferences, can EARS design a more aligned reward function than directly prompting an LLM?	
▶ **Yes** in $4/6$ settings. We find large gains where direct-prompting produces poorly aligned reward functions, and results are comparable where all methods produced well-aligned ones.	Table 2
Given an objective specification in natural language, does EARS design a more aligned reward function than directly prompting an LLM with the same input?	
▶ **Yes** in $5/6$ settings.	Table 4
Ablations	
Does sampling imagined trajectories from feature space produced by EARS's Stage 1 yield better pairs than sampling from the raw observation space?	
▶ **Yes**; sampling imagined trajectories from the raw observation space and eliciting preferences over them yields significantly less aligned reward functions.	Table 3
Does learning from preferences over real trajectories require fewer preferences than learning over EARS' imagined trajectories?	
▶ **No**: EARS matches or outperforms the data-efficiency of two methods that learn from real trajectories, which can be costly to acquire from the real environment.	Figure 2

Table 5: **Summary of experiments and findings.** Each row reports one experiment, the methods it is compared against, and the question it answers. MAIN COMPARISONS evaluate EARS against methods that, like EARS, require no environment sampling; ABLATIONS isolate the contribution of Stage 1 and compare against methods that do sample real trajectories.

5 Conclusion

We introduced EARS, a method for specifying reward functions from preferences without sampling trajectories from the environment—which can be prohibitively slow or unsafe. Across three sequential decision-making domains, EARS produces reward functions more aligned with the ground-truth reward function than the only existing alternatives that avoids environment sampling: directly prompting an LLM to generate a reward function. This holds in both evaluation settings we consider—when preferences are generated from a ground-truth reward function, and when they are labeled by an LLM conditioned on the kind of natural language specification a careful human stakeholder could readily provide without ever writing a reward function.. EARS also matches or outperforms methods that learn from preferences labeled over real trajectories.

Several directions for future work would extend the applicability of EARS. The setting we evaluate in Section 4.5—where a stakeholder supplies a natural language specification of the task objectives and an LLM labels preferences over imagined trajectories—requires no human in the loop and is readily practical. A natural extension is to evaluate EARS in a setting where preferences are labeled by human stakeholders rather than an LLM or a ground-truth reward function. This raises open questions our current evaluation cannot answer: whether stakeholders can meaningfully reason about imagined trajectories that may not be physically realizable. A user study with domain experts would address these questions. While we show that LLMs can design reward functions by labeling preferences over imagined trajectories given a natural language specification, what kinds of specifications human stakeholders *can* or *should* provide remain important open questions.

References

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Erdem Biyik, Nima Anari, and Dorsa Sadigh. Batch active learning of reward functions from human preferences. *ACM Transactions on Human-Robot Interaction*, 13(2):1–27, 2024.

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

Gui Citovsky, Giulia DeSalvo, Claudio Gentile, Lazaros Karydas, Anand Rajagopalan, Afshin Rostamizadeh, and Sanjiv Kumar. Batch active learning at scale. *Advances in Neural Information Processing Systems*, 34:11933–11944, 2021.

Ryan Cosner, Maegan Tucker, Andrew Taylor, Kejun Li, Tamas Molnar, Wyatt Ubelacker, Anil Alan, Gábor Orosz, Yisong Yue, and Aaron Ames. Safety-aware preference-based learning for safety-critical control. In *Learning for Dynamics and Control Conference*, pp. 1020–1033. PMLR, 2022.

Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. Rlhf workflow: From reward modeling to online rlhf. *arXiv preprint arXiv:2405.07863*, 2024.

Vikranth Dwaracherla, Seyed Mohammad Asghari, Botao Hao, and Benjamin Van Roy. Efficient exploration for llms. *arXiv preprint arXiv:2402.00396*, 2024.

Ian Fox, Joyce Lee, Rodica Pop-Busui, and Jenna Wiens. Deep reinforcement learning for closed-loop blood glucose control. In *Machine Learning for Healthcare Conference*, pp. 508–536. PMLR, 2020.

Javier Garcia and Fernando Fernández. Safe exploration of state and action spaces in reinforcement learning. *Journal of Artificial Intelligence Research*, 45:515–564, 2012.

Stephane Hatgis-Kessell, W Bradley Knox, Serena Booth, Scott Niekum, and Peter Stone. Influencing humans to conform to preference models for rlhf, 2025. *URL https://arxiv. org/abs/2501.06416*.

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

Kihyun Kim, Jiawei Zhang, Asuman Ozdaglar, and Pablo A Parrilo. A unified linear programming framework for offline reward learning from human demonstrations and feedback. *arXiv preprint arXiv:2405.12421*, 2024.

W Bradley Knox, Stephane Hatgis-Kessell, Serena Booth, Scott Niekum, Peter Stone, and Alessandro Allievi. Models of human preference for learning reward functions, june 2022. *URL https://arxiv. org/abs/2206.02231 v3*.

Varun Kompella, Roberto Capobianco, Stacy Jong, Jonathan Browne, Spencer Fox, Lauren Meyers, Peter Wurman, and Peter Stone. Reinforcement learning for optimization of covid-19 mitigation policies. *arXiv preprint arXiv:2010.10560*, 2020.

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

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

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

Yi Liu, Gaurav Datta, Ellen Novoseller, and Daniel S Brown. Efficient preference-based reinforcement learning using learned dynamics models. *arXiv preprint arXiv:2301.04741*, 2023.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

Chiara Dalla Man, Francesco Micheletto, Dayu Lv, Marc Breton, Boris Kovatchev, and Claudio Cobelli. The UVA/PADOVA type 1 diabetes simulator. *Journal of Diabetes Science and Technology*, 8(1):26–34, January 2014. ISSN 1932-2968. DOI: 10.1177/1932296813514502. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4454102/.

Calarina Muslimani, Kerrick Johnstonbaugh, Suyog Chandramouli, Serena Booth, W Bradley Knox, and Matthew E Taylor. Towards improving reward design in rl: A reward alignment metric for rl practitioners. *arXiv preprint arXiv:2503.05996*, 2025.

Ian Osband, Zheng Wen, Seyed Mohammad Asghari, Vikranth Dwaracherla, Morteza Ibrahimi, Xiuyuan Lu, and Benjamin Van Roy. Epistemic neural networks. *Advances in Neural Information Processing Systems*, 36:2795–2823, 2023.

Aldo Pacchiano, Aadirupa Saha, and Jonathan Lee. Dueling rl: reinforcement learning with trajectory preferences. *arXiv preprint arXiv:2111.04850*, 2021.

Alexander Pan, Kush Bhatia, and Jacob Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models, 2022. *URL https://arxiv. org/abs/2201.03544*, 2022.

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

Cathy Wu, Abdul Rahman Kreidieh, Kanaad Parvate, Eugene Vinitsky, and Alexandre M Bayen. Flow: A modular learning framework for mixed autonomy traffic. *IEEE Transactions on Robotics*, 38(2):1270–1286, 2021.

Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. Text2reward: Reward shaping with language models for reinforcement learning. *arXiv preprint arXiv:2309.11489*, 2023.

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. Language to rewards for robotic skill synthesis. *arXiv preprint arXiv:2306.08647*, 2023.

Tan Zhi-Xuan, Micah Carroll, Matija Franklin, and Hal Ashton. Beyond preferences in ai alignment. *Philosophical Studies*, 2024. DOI: 10.1007/s11098-024-02249-w. arXiv:2408.16984.

Table 4: **EARS with natural language input compared to prompting baselines:** We evaluate EARS with preferences labeled by an LLM given a natural language specification generated by the ground-truth reward function against direct prompting baselines that generate reward functions from the same specification. We evaluate with two specifications outlined in Section 4.3: (a) Realistic Reward Description and (b) Privileged Reward Description. See Table 2 for more details.

(a) Realistic Reward Description

	Pandemic		Glucose		Traffic	
Variant	r_1	r_2	r_1	r_2	r_1	r_2
Direct Prompting	$-.37\pm.10$	$.68\pm.06$	$.27\pm.29$	$.79\pm.14$	$\mathbf{.27\pm.04}$	$.94\pm.01$
EARS	$\mathbf{.43\pm.16}$	$\mathbf{.74\pm.02}$	$\mathbf{.85\pm.10}$	$\mathbf{.89\pm.04}$	$-.03\pm.14$	$\mathbf{.95\pm.02}$

(b) Privileged Reward Description

	Pandemic		Glucose		Traffic	
Variant	r_1	r_2	r_1	r_2	r_1	r_2
Direct Prompting	$-.14\pm.14$	$.58\pm.19$	$.21\pm.17$	$.84\pm.12$	$.17\pm.01$	$\mathbf{1.00\pm.00}$
EARS	$\mathbf{.32\pm.08}$	$\mathbf{.71\pm.07}$	$\mathbf{.84\pm.00}$	$\mathbf{.95\pm.00}$	$\mathbf{.20\pm.20}$	$.97\pm.03$

Supplementary Materials

The following content was not necessarily subject to peer review.

6 Evaluating the learned reward function.

To evaluate the alignment of the learned reward function with respect to a ground-truth reward function, we compute the TAC between the two as outlined in Section 4.1. An alternative method to evaluate $\hat{r}$ could be to compute the the set of optimal policies with respect to $\hat{r}$ and evaluate the average performance of those polices under the r: $\frac{1}{|\pi^*_{\hat{r}}|}\sum_{\pi\in\pi^*_{\hat{r}}} J_r(\pi)$. We avoid this evaluation methodology because, for the environments we consider, training any policy is computationally expensive and, due to the long task horizons and continuous state spaces, deriving an optimal policy is intractable. For a sufficiently large and diverse set of evaluation trajectories, $\mathrm{TAC}(\hat{r}, r)$ provides a strong measure of alignment; unlike empirical policy performance, it evaluates how the reward functions rank all trajectories, rather than focusing only on the observed highest-ranked ones. Broadly, the focus of our work is on producing aligned reward functions, which TAC evaluates directly, rather than reward functions that can train policies.

7 Labeling synthetic preferences over imagined trajectories

We evaluate EARS in Section 4.4 by assuming access to a synthetic preference labeler that labels preferences according to the ground-truth reward function. These preferences are labeled over imagined trajectories rather than real trajectories sample from an environment, and the ground-truth reward function is defined over transitions sampled from the real environment (e.g., $r(s, a, s')$). Therefore, to enable learning from synthetic preferences over imagined trajectories, we find r_m that best satisfies the ranking induced by r over all trajectories in $\mathcal{T}_{\text{eval}}$. Following the Bradley–Terry formulation, we model pairwise preferences as

$$P(\Phi_\tau \succ \Phi_{\tau'} \mid r_m) = \sigma\big(r_m(\Phi_\tau) - r_m(\Phi_{\tau'})\big),$$

and learn r_m by minimizing the cross-entropy loss against the ranking induced by r:

$$\mathcal{L}(r_m; \mathcal{T}_{\text{eval}}) = -\sum_{\tau,\tau'\in\mathcal{T}_{\text{eval}}} (1 - \mu_{\tau,\tau'}) \log P(\Phi_\tau \succ \Phi_{\tau'} \mid r_m) + \mu_{\tau,\tau'} \log P(\Phi_\tau \prec \Phi_{\tau'} \mid r_m), \tag{5}$$

where $\mu_{\tau,\tau'} = \mathbb{1}[r(\tau') > r(\tau)]$ encodes the ground-truth, noiseless preference under r.

To avoid arbitrary scaled of the mapped ground-truth reward function, we then scale the mapped ground-truth reward function such that the mean preference probability under the Bradley-Terry model for preferences over all trajectories in $\mathcal{T}_{\text{eval}}$ is 0.9. Synthetic preferences over imagined trajectories can then be labeled with the mapped ground-truth reward function.

8 Implementation Details for LTRRT and Offline-RLHF

We compare EARS against two preference-learning baselines that operate directly on environment observations rather than on LLM-generated reward features: *Learning To Rank Real Trajectories* (LTRRT) and *Offline-RLHF*. Both baselines are trained and evaluated entirely offline from a fixed pool of pre-collected trajectories. The two baselines are identical in every respect—trajectory pool, preference labeling, reward-model architecture, optimizer, and evaluation—and differ *only* in how a trajectory is encoded into the fixed-length feature vector consumed by the reward model. This isolates the effect of the trajectory representation.

8.1 Preference-pair construction

For each environment we use a fixed dataset of 450 pre-collected trajectories, which are sampled from various checkpoints when training a policy with the ground-truth reward functions. Each

trajectory is a sequence of (s, a, s') transitions. We uniformly sample pairs of trajectories from the fixed dataset of 900 trajectories.

8.2 Trajectory representations (the only difference between baselines)

Let $\phi(s_t, a_t, s_{t+1}) \in \mathbb{R}^d$ denote the reward feature vector for a single transition (s_t, a_t, s_{t+1}).

LTRRT (Learning To Rank Real Trajectories). A trajectory is summed over its transitions into a single d-dimensional vector, $\Phi^{\text{LTRRT}} = \sum_{t=0}^{|\tau|-1} \phi(s_t, a_t, s_{t+1}) \in \mathbb{R}^d$, and then min–max normalized per dimension using precomputed per-feature ranges $[\ell_m, h_m]$, i.e. $\Phi_m^{\text{LTRRT}} \leftarrow (\Phi_m^{\text{LTRRT}} - \ell_m)/(h_m - \ell_m)$. The model input dimension is therefore d, independent of trajectory length.

Offline-RLHF. A trajectory is encoded by concatenating the per-transition feature vectors across all transitions, $\Phi^{\text{RLHF}} = [\phi(s_0, a_0, s_1); \phi(s_1, a_1, s_2); \ldots; \phi(s_{|\tau|-1}, a_{|\tau|-1}, s_{|\tau|})] \in \mathbb{R}^{|\tau| \cdot d}$. These features are *not* normalized. This representation preserves the full per-transition feature trace, at the cost of a much higher input dimension.

8.3 Training details

Both baselines learn a model $f_\theta(\cdot)$ implemented as a neural network. The network has 2 hidden layers of width 64 with $\tanh$ activations, followed by a linear layer to a single scalar output. The input dimension equals the representation dimension of the corresponding baseline. The two baselines use exactly the same architecture and hyperparameters; only the input dimension differs.

The reward model is trained by minimizing the pairwise binary cross-entropy loss from Eq. 2. We use Adam with learning rate 10^{-3}, weight decay 0, for 1000 epochs.

9 Ground-truth reward function descriptions

The ground-truth reward functions we use as alignment targets are given below for all environments:

$$r_1 = \begin{cases} 0, & \text{if collision} \\ \eta_1 c_1 + \eta_2 c_2 + \eta_3 c_3, & \text{otherwise} \end{cases}$$

$$c_1 = \frac{\max\big(v^\star \sqrt{N} - \|\mathbf{v} - v^\star \mathbf{1}\|_2,\ 0\big)}{v^\star \sqrt{N}},$$

$$c_2 = \sum_{i:\, v_i^{\text{ego}} > 0} \min\left(\frac{h_i / v_i^{\text{ego}} - t_{\min}}{t_{\min}},\ 0 \right),$$

$$c_3 = \min\left(\alpha - \tfrac{1}{M} \sum_{j=1}^{M} |a_j|,\ 0 \right).$$

Traffic Control Ground-truth reward function r_1. $\mathbf{v} \in \mathbb{R}^N$ are the speeds of all N vehicles; $v^\star$ is the target velocity; v_i^{ego} and h_i are the speed and leader headway of the i-th RL vehicle; $\mathbf{a} \in \mathbb{R}^M$ are the accelerations of the M RL vehicles; $t_{\min} = 1$ s; $\alpha = 0.1$; $(\eta_1, \eta_2, \eta_3) = (1.0,\ 0.1,\ 1.0)$.

$$r_2 \;=\; \frac{1}{N}\sum_{i=1}^{N} v_i.$$

Traffic Control Ground-truth reward function r_2. v_i is the speed of the i-th vehicle and N is the total number of vehicles in the simulator.

$$r_1 \;=\; -10\,C_t \;-\; 10\,\mathbb{1}[\,s_t - s_{t-1} = 1\,]\left(\frac{\min(\rho_{t-1} - 0.005,\,0)}{0.005}\right)^2 \;-\; 0.1\,\frac{s_t^{1.5}}{4^{1.5}} \;-\; 0.02\,|s_t - s_{t-1}|.$$

Pandemic Mitigation Ground-truth reward function r_1. $s_t \in \{0,1,2,3,4\}$ is the regulation stage at step t; C_t is the population-mean fraction of critical cases at step t; $\rho_{t-1} = I_{t-1} + C_{t-1} + D_{t-1}$ is the sum of the infected, critical, and dead population fractions at the previous step; $\mathbb{1}[\cdot]$ is the indicator function.

$$r_2 \;=\; -10\,C_t \;-\; 0.1\,\frac{s_t^{1.5}}{4^{1.5}} \;-\; 0.02\,|s_t - s_{t-1}|.$$

Pandemic Mitigation Ground-truth reward function r_2. $s_t \in \{0,1,2,3,4\}$ is the regulation stage at step t; C_t is the population-mean fraction of critical cases at step t.

$$r_1 \;=\; \begin{cases} -10^{13}, & \text{if terminated} \\ -10\left[\,3.5506\big(\ln(\max(b_t,1))^{0.8353} - 3.7932\big)\right]^2 \;-\; 10\,u_t, & \text{otherwise} \end{cases}$$

Glucose Monitoring Ground-truth reward function r_1. b_t is the most recent blood-glucose reading (mg/dL); u_t is the most recent insulin dose; the terminated case corresponds to patient death.

$$r_2 \;=\; \begin{cases} -10^{13}, & \text{if terminated} \\ -0.32\,u_t \;-\; \mathbb{1}[\,b_t < 70\,]\,\dfrac{10\cdot 1350}{12\cdot 24\cdot 365}, & \text{otherwise} \end{cases}$$

Glucose Monitoring Ground-truth reward function r_2. b_t is the most recent blood-glucose reading (mg/dL); u_t is the most recent insulin dose; $\mathbb{1}[b_t < 70]$ flags hypoglycemia; the terminated case corresponds to patient death.

As an input to the Direct Prompting baseline, and to the LLM-preference labeler that is used for EARS in Section 4.5, we use a natural language description of the ground-truth reward function. The Privileged Reward Description, which we hypothesize is likely more specific than a human stakeholder could reasonably provide, is given in Figures 3 and 5 for Pandemic Mitigation, Figures 7 and 9 for Glucose Monitoring, and Figures 11 and 13 for Traffic Control.

The Realistic Reward Description, which we hypothesize is a specification a human stakeholder could reasonably provide, is given in Figures 4 and 6 for Pandemic Mitigation, Figures 8 and 10 for Glucose Monitoring, and Figures 12 and 14 for Traffic Control.

The reward function should aggregate four distinct penalty components to guide pandemic management policies effectively. First, it should significantly penalize the proportion of the population currently in a critical infection state to prioritize public health. Second, it must apply a specific political penalty for increasing regulation stringency when the infection rate is negligible or very low. Third, it should discourage the maintenance of strict regulation stages, imposing a penalty that scales with the severity of the current restrictions. Finally, the function should penalize the magnitude of changes in regulations between time steps to encourage smooth and stable policy transitions.
The tradeoffs between the features of the reward function are as follows:

- Increasing the proportion of critical cases in the population by 0.01 units is equivalent to increasing the normalized stage severity penalty by 1.0 unit.
- Increasing the magnitude of the regulation stage change by 1.0 unit is equivalent to increasing the normalized stage severity penalty by 0.2 units.
- Increasing the calculated political penalty factor by 1.0 unit is equivalent to increasing the proportion of critical cases in the population by 1.0 unit.

Figure 3: Pandemic Mitigation Privileged Reward Description for ground-truth reward function r_1

What outcomes should the agent try to achieve?

- Keep the patient's blood glucose levels within a healthy target range. The primary goal is to maintain stable blood sugar levels that mimic a healthy person's pancreas.
- Administer insulin efficiently. The agent should try to achieve the target glucose levels using the least amount of medication necessary to get the job done.

What outcomes should the agent avoid?

- The agent must absolutely avoid any state that leads to the termination of the simulation or the death of the patient. This is the worst-case scenario.
- Avoid letting blood glucose levels get dangerously high (hyperglycemia) or dangerously low (hypoglycemia). The further the blood sugar drifts from the ideal target, the worse it is for the patient.
- Avoid "insulin stacking" or using excessive amounts of the drug when it isn't strictly needed, as this increases the risk of a crash later.

Describe how the agent should trade off between different objectives or outcomes.

- The most critical priority is preventing the simulation from ending in failure or patient death; this outweighs everything else.
- When balancing blood sugar management against medication usage, prioritize keeping the blood sugar in the safe range. However, if two strategies result in similar blood glucose levels, prefer the one that uses less insulin to minimize the burden on the patient.
- Being slightly outside the target range is bad, but being extremely far outside the target range (very high or very low) is significantly worse and should be penalized much more heavily.

Figure 8: Glucose Monitoring Realistic Reward Description for ground-truth reward function r_1

What outcomes should the agent try to achieve?

- **Minimize severe health consequences:** The primary goal is to keep the number of critical infection cases as low as possible. We want to prevent the healthcare system from being overwhelmed.
- **Maintain normalcy/freedom:** The agent should aim to keep the restriction level as low as possible. If it is safe to do so, the preference is always for looser regulations rather than strict lockdowns.
- **Policy consistency:** The agent should try to keep the chosen restriction level stable over time. People need to be able to plan their lives, so we want to see long stretches of the same stage rather than constant changes.

What outcomes should the agent avoid?

- **High numbers of critical patients:** Avoid situations where a large portion of the population falls critically ill.
- **Severe lockdowns:** Avoid using the highest levels of government intervention unless it is the only option left.
- **Erratic "flip-flopping":** Avoid changing the restriction stage abruptly or frequently. For example, don't jump from no restrictions to a full lockdown and back again within a short period.
- **Unjustified tightening:** Specifically avoid increasing the restriction level when the current number of infections is actually low. It looks bad politically and socially to impose harsher rules when the virus isn't actively surging.

Describe how the agent should trade off between different objectives or outcomes.

- While public health is critical, the agent needs to carefully balance it against the economic and social cost of lockdowns. Do not choose a high restriction level just to prevent a very small number of cases; only use strict measures if the critical cases are becoming significant.
- Prioritize consistency over immediate reaction. Even if a slight change might be optimal for infection numbers, it is often better to stay the course to avoid confusing the public.
- Be very hesitant to increase restrictions if the infection rate is currently low. The "cost" of adding restrictions feels much higher to the population when they don't see an immediate threat, so the agent should really resist doing that.

Figure 4: Pandemic Mitigation Realistic Reward Description for ground-truth reward function r_1

The reward function should guide the agent to balance public health outcomes with the economic and social costs of restrictions and policy instability. It must penalize the presence of critical infection cases within the population to prioritize health safety. Simultaneously, the function should penalize the severity of the current regulation stage, encouraging the agent to keep restrictions as low as possible. Additionally, it must discourage frequent or drastic changes to regulations by penalizing the difference in stages between time steps to ensure smooth policy transitions.
The tradeoffs between these features interact as follows: increasing the proportion of the population in a critical condition by 1% (0.01) is equivalent to the penalty of imposing the maximum level of regulation severity. Furthermore, increasing the critical population proportion by just 0.2% (0.002) results in a penalty equivalent to changing the regulation stage by a single level. Consequently, the penalty for maintaining the maximum regulation severity is equivalent to the penalty accumulated by changing the regulation stage 5 times.

Figure 5: Pandemic Mitigation Privileged Reward Description for ground-truth reward function r_2

What outcomes should the agent try to achieve?

- Keep the number of people who are critically ill as low as possible. The primary goal is to save lives and prevent the healthcare system from getting overwhelmed, so focus on minimizing the number of critical infection cases.
- Keep society as open as possible. Whenever it is safe to do so, the agent should aim for the lowest level of restrictions (like Stage 0 or Stage 1) to minimize the impact on the economy and people's daily lives.

What outcomes should the agent avoid?

- Avoid situations where the number of severe infections spikes uncontrollably. We don't want to reach a point where hospitals cannot cope with the influx of patients.
- Avoid erratic or "flip-flopping" policies. The agent should avoid drastically changing the restriction level back and forth over short periods (e.g., going from Level 1 to Level 4 and back to Level 1 in a few days), as this causes confusion and instability.

Describe how the agent should trade off between different objectives or outcomes.

- Public health is the priority, but not at any cost. If critical cases are rising and threatening the healthcare system, the agent must accept higher restrictions to bring numbers down. However, if the health situation is stable or critical cases are low, the agent should prioritize lowering restrictions to relieve the social and economic burden.
- Regarding policy changes: The agent should favor stability. It is better to maintain a slightly higher restriction level for a bit longer than to lower it and immediately have to raise it again. Only make large jumps in restrictions if there is an extreme emergency; otherwise, prefer steady, predictable changes.

Figure 6: Pandemic Mitigation Realistic Reward Description for ground-truth reward function r_2

The reward function should serve to maintain healthy blood glucose levels while minimizing insulin usage, with a strict priority on preventing patient death. If the simulation terminates, a very large negative penalty must be applied immediately. Otherwise, the function should calculate a risk metric based on the current blood glucose level using a non-linear transformation that penalizes deviations from the target range. This risk score is then combined with a penalty proportional to the amount of insulin administered to form a negative cost function. The final reward is the negative sum of the glucose risk and the insulin penalty. In terms of tradeoffs, increasing the squared transformed glucose risk value by one unit is equivalent to increasing the insulin dosage by one unit.

Figure 7: Glucose Monitoring Privileged Reward Description for ground-truth reward function r_1

The reward function should serve as an economic optimizer that treats the agent's actions and physiological state as financial costs to be minimized. It must assign a catastrophic negative value if the patient does not survive, ensuring that maintaining the patient's life is the paramount constraint. Routine usage of insulin should be penalized proportionally to the amount administered, reflecting the direct financial cost of the medication. Additionally, the function should apply a discrete penalty whenever blood glucose levels drop below a critical safety threshold, simulating the probabilistic cost of emergency medical care or hospitalization. The overall objective is to maximize the negative sum of these expenses, encouraging the agent to balance medication costs against the financial risks of severe health complications.
In terms of feature tradeoffs, the reward is composed of negative costs associated with insulin usage and hypoglycemic events. Specifically, the financial penalty for administering 1 unit of insulin is approximately 2.5 times greater than the penalty incurred for a single time-step of hypoglycemia. Therefore, avoiding a single instance of low blood glucose is equivalent to saving approximately 0.4 units of insulin.

Figure 9: Glucose Monitoring Privileged Reward Description for ground-truth reward function r_2

What outcomes should the agent try to achieve?

- The primary goal is to minimize the overall financial burden of treating the patient. I want the agent to find a strategy that results in the lowest possible total cost over the treatment period.
- The agent should manage the patient's condition efficiently, using resources only when necessary to keep costs down.

What outcomes should the agent avoid?

- Most importantly, the agent must strictly avoid any situation where the patient dies or reaches a fatal state. This is unacceptable.
- Avoid letting the patient's blood sugar drop too low (hypoglycemia). This is crucial because low blood sugar usually requires expensive emergency interventions or hospital visits.
- Avoid wasting insulin. Every unit of insulin administered costs money, so the agent shouldn't administer it unless the benefit outweighs the direct cost of the drug.

Describe how the agent should trade off between different objectives or outcomes.

- Survival is the hard constraint; never let the patient reach a terminal state regardless of cost.
- Beyond survival, treat medical decisions as financial calculations. The agent should balance the direct cost of giving insulin against the potential future cost of emergency care.
- It is acceptable to allow higher blood sugar levels if bringing them down would cost more in medication than it saves in avoided hospital fees, provided the patient stays alive. The strategy should essentially be: "Spend the least amount of money possible while keeping the patient alive."

Figure 10: Glucose Monitoring Realistic Reward Description for ground-truth reward function r_2

The reward function should encourage efficient, safe, and smooth traffic flow by combining a system-wide velocity objective with specific penalties for unsafe or aggressive driving behaviors. It must calculate a positive, normalized score based on how closely the speeds of all vehicles in the network align with a desired target velocity. Simultaneously, the function should enforce safety by applying a penalty whenever an RL agent's time-headway (distance normalized by speed) relative to its leader falls below a minimum safe threshold. To ensure passenger comfort and fuel efficiency, it must also deduct a penalty if the average absolute acceleration of the agents exceeds a defined comfort limit. In the event of a collision or system failure, the function should immediately disregard these components and return a reward of zero.
The features of the reward function interact with specific tradeoffs between system efficiency, safety spacing, and driving smoothness. The global velocity score and the acceleration penalty carry equal weight (1:1); therefore, increasing the normalized velocity score by 1 unit is equivalent to reducing the excess acceleration violation (above the comfort threshold) by 1 unit. However, the penalty for unsafe time-headway is weighted significantly lower (at 10% of the other terms), meaning that increasing the velocity score by just 0.1 units is equivalent to improving the normalized time-headway violation metric by a full 1.0 unit.

Figure 11: Traffic Control Privileged Reward Description for ground-truth reward function r_1

What outcomes should the agent try to achieve?

- **Efficient Traffic Flow:** The primary goal is to keep the entire fleet moving well. I want the vehicles to try to match the target speed of the highway so that traffic doesn't bottle up. It's important that the whole system flows nicely, not just individual cars speeding up.
- **Safe Following Distances:** Vehicles should maintain a safe gap behind the car in front of them. They need to leave enough room (time gap) to react safely if the lead car slows down.
- **Passenger Comfort:** The vehicles should drive smoothly. This means accelerating and decelerating in a way that feels natural and isn't jarring for passengers.

What outcomes should the agent avoid?

- **Collisions and Failures:** The most obvious one is that the cars must not crash into each other or go off the road. This is a critical failure.
- **Tailgating:** Avoid getting too close to the vehicle ahead. Following too closely is unsafe and stressful.
- **Jerky Driving:** Avoid sudden, erratic movements. The agents shouldn't be slamming on the brakes or flooring the gas pedal unless absolutely necessary to avoid a crash.

Describe how the agent should trade off between different objectives or outcomes.

- Safety is the absolute baseline. If a collision or a major failure happens, the attempt counts for nothing—it should receive no reward at all (or a zero score).
- As long as no accidents occur, you should balance the other goals. I want the cars to be efficient and reach the target speed, but not if it means driving aggressively (jerky) or following unsafely (tailgating). Try to find a sweet spot where traffic moves fast but the ride remains smooth and the gaps between cars stay safe.

Figure 12: Traffic Control Realistic Reward Description for ground-truth reward function r_1

The reward function should prioritize maximizing the overall traffic flow efficiency across the entire network. To achieve this, it should calculate a reward based on the speeds of all vehicles present in the environment, encompassing both the reinforcement learning agents and the uncontrolled background traffic. The specific signal provided to the agent should be the average velocity of this collective fleet. This design encourages the agents to cooperate in a way that prevents congestion and maintains high speeds for every driver on the road. Since the reward function consists solely of the global average speed, there are no tradeoffs between competing objectives like safety or energy consumption; consequently, increasing the speed of any single vehicle by a specific amount contributes exactly the same value to the total reward as increasing the speed of any other vehicle by that same amount.

Figure 13: Traffic Control Privileged Reward Description for ground-truth reward function r_2

What outcomes should the agent try to achieve?

- Keep traffic moving as fast as possible. The main goal is to have a high driving speed for every car on the road, not just the autonomous ones.
- Ensure the overall flow of traffic is efficient. When the AVs merge, the cars on the main highway should be able to keep driving quickly without having to slow down significantly.
- Improve the throughput of the whole road network. I want to see the fleet acting in a way that helps everyone get to their destination faster.

What outcomes should the agent avoid?

- Avoid causing traffic jams or bottlenecks. The AVs shouldn't do anything that causes a "shockwave" of braking behind them.
- Avoid situations where cars are stuck or crawling along. Any situation where the average speed of the cars drops to near zero is bad.
- Avoid collisions, obviously, but also avoid forcing human drivers on the highway to slam on their brakes to let the AVs in, as this creates congestion.

Describe how the agent should trade off between different objectives or outcomes.

- The most important thing is the collective speed of all cars. It is acceptable for one specific autonomous vehicle to slow down or wait a moment on the ramp if doing so prevents a traffic jam on the main highway.
- Prioritize the "big picture" flow over the speed of any single vehicle. If slowing down one car allows ten others to maintain a high speed, the agent should do that. The goal is maximizing the speed of the group, not just the individual.

Figure 14: Traffic Control Realistic Reward Description for ground-truth reward function r_2