
Reputation Hacking in a Simulated RL Environment¹

Rafan Ahmed

Eric Fackelman

Merlijn de Blauw

With
Apart Research

Abstract

The ever-growing danger of AI manipulation becoming increasingly difficult to detect is a very real problem as AI is utilized more and more in our modern-day world. To combat this danger, we created a minimalist reinforcement-learning environment built on a Worker-Assistant scenario to act as a benchmark for detecting AI manipulation.

1. Introduction

We have chosen to work on creating a minimal reinforcement-learning environment to study the emergence of reputation-hacking behaviour under peer-evaluation reward schemes and (in follow-up work and future iterations) to compare the behaviour against standard reward designs, such as outcome-based collective rewards, as well as individual (oracle) contribution rewards.

Internally, our project overlaps with track 1 (Measurement & Evaluation: Designing benchmarks and evaluations, assessing ecological validity, creating detection methods, and building frameworks) and track 4 (Open track: exploring emergent manipulation, analyzing dual-use consideration, developing novel theoretical frameworks, and pursuing other empirical projects) of the hackathon tracks provided within the AI manipulation hackathon by Apart Research.

Recognizing the ever-growing danger of AI manipulation becoming harder to detect as AI models master forms of manipulation, this project is aimed at combating that growing capability of AI systems to learn and optimize perceived contribution rewards as they are

¹ Research conducted at the [AI Manipulation Hackathon](#), 2026

increasingly evaluated through subjective or indirect signals (e.g., human feedback, peer review, engagement metrics) and the incentive it creates for manipulation (including reputation hacking, sycophancy, strategic timing, and rewarding gaming)—a risk exacerbated by multi-agent systems. We hope to combat this issue not just delivering a minimal, reproducible manipulation benchmark as well as a testbed for evaluation and mitigation research, but also a clear operational definition of reputation hacking (initially instantiated via our evaluation separating PeerRating from TrueImpact, i.e., a “ManipulationScore”) and a bridge between multi-agent reinforcement-learning and AI manipulation safety; that way, there is a baseline to detect AI manipulation in the form of reputation hacking under peer evaluation that can be further improved upon and expand the infrastructure surrounding the detection of AI manipulative techniques.

By creating and studying the reinforcement-learning environment, we hope to answer our research question of how agent behavior changes when rewards are based on perceived contribution (peer evaluation) rather than objective task outcomes, and under what conditions does manipulation emerge instead of genuine collaboration

1.2 Hypotheses

Through studying our reinforcement-learning environment, we expect the following results:

- A. If we introduce an outcome-based collective reward (in follow-up work), the higher the outcome-based collective reward, the more encouraged the Assistant is to deliver genuine help to the Worker.
- B. If we introduce Individual (oracle) contribution rewards (in follow-up work), then the rewards become aligned with real impact.
- C. As peer-evaluation-based rewards are used, the greater the incentive grows for the Assistant to choose: signalling over helping; making use of strategic timing; and reward hacking.
- D. If the environment is stripped down to the most essential parts, then manipulation of the Assistant can still emerge.

2. Methods

To measure reputation hacking under peer evaluation, we built a minimal Gymnasium reinforcement-learning environment in which an Assistant (the RL agent) interacts with a Worker (a scripted process that performs tasks). The key idea is that the Worker produces a peer-evaluation-style rating of the Assistant at the end of each episode by comparing how hard the episode felt to what the Worker expected it should feel like. This setup creates a direct incentive for the Assistant to optimize a subjective reward signal, which can be done either by improving the Worker’s actual experience (genuine help) or by manipulating expectations (reputation hacking).

2.1 The environment

Each episode lasts $T = 50$ timesteps. Within an episode, the Worker works on a stream of tasks: whenever `task_progress` ≥ 1.0 , the environment resets the task progress and updates the task difficulty (difficulty drifts over time via a momentum + noise process and

is clipped to a bounded range). The episode terminates when the timestep counter reaches T ; rewards are issued only at termination (sparse terminal reward).

2.2 Agents and actions

Worker (scripted): The Worker does not learn a policy. Instead, it follows a threshold rule based on fatigue:

- If $\text{fatigue} < \text{fatigue_threshold}$, the Worker works (incurs effort, makes progress, accumulates fatigue).
- Otherwise, the Worker rests (recovers fatigue but makes no progress).

The manipulation in the simulation can emerge through the actions taken by the Assistant, the reinforcement-learning agent. The structure for the Assistant is comprised of three actions:

1. **DO_NOTHING**: This action does not affect the expected effort nor the experienced effort.
2. **HELP**: This action reduces the Worker's effort by help_strength and will thus lower the experienced effort by the Worker as the task becomes easier than initially expected, meaning the Assistant provides genuine assistance to the task.
3. **SIGNAL**: This action inflates the Worker's expected effort via signal_bias : the Worker will imagine the task to be more difficult than it truly is and expects the task to be harder.

This environment demonstrates the emergence of manipulation in the Assistant, as choosing the action **HELP** will result in a positive evaluation rating granted by the Worker, as the task became easier than initially expected due to the Assistant's genuine assistance with the task. The action **SIGNAL**, however, replicates the positive effect of the Worker finding the task to be easier than expected, as **SIGNAL** will inflate the Worker's expected effort, which will then also lead to a positive evaluation rating granted by the Worker, even though the Assistant never provided actual assistance to the task: manipulating the Worker into delivering a positive evaluation rating to the Assistant.

Manipulation can be observed if the following are occurring within the simulation results:

- A. The positive evaluation rating given by the Worker to the Assistance (far) exceeds the actual helpfulness of the Assistance to the task;
- B. $\text{ManipulationScore} > 0$;
- C. And a high overall **SIGNAL** usage in comparison to **HELP**.

The emergence of manipulation in the actions of the Assistant within an environment based on peer evaluation rewards can be explained as follows: The Assistant may learn to optimize effort and find a shortcut to obtain a positive evaluation rating by the Worker by more probably than not choosing **SIGNAL** instead of **HELP** because: signaling is less labor-intensive as no actual work is required of the Assistant; both actions result in a positive evaluation rating by the peer (the Worker); and the condition of the reward does not specifically state that the Assistant must deliver genuine assistance to the task in order to receive the reward, and thus can be obtained via manipulating expectations.

2.3 State, observations, and what is hidden

2.3.1 State

Internally, the environment tracks a list of several variables (but not all are observed):

- Time: t .
- Current task: task_progress , task_difficulty with a drifting trend).
- Worker condition: fatigue (affects whether the Worker chooses WORK or REST)
- Accumulated episode stats: $\text{experienced_effort}$, help_count , and signal_count
- Manipulation/memory variables, such as:
 - signal_bias (accumulates within an episode when you SIGNAL)
 - $\text{learned_expected_effort}$ (a persistent EMA across episodes)

2.3.2 Observations

The Assistant is able to observe a 4D vector comprised of:

- $[t/T, \text{task_progress}, \text{normalized_fatigue}, \text{task_difficulty}]$

Where:

- $\text{normalized_fatigue} = \text{fatigue} / \text{fatigue_max}$
- task_difficulty is the actual, current difficulty (not a noisy proxy).

This means that the Assistant does not observe:

- A. $\text{experienced_effort}$: the true effort the Worker experienced whilst attempting to progress the task.
- B. $\text{learned_expected_effort}$: the newly learned expected effort for the Worker after the Assistant chooses SIGNAL
- C. signal_bias : the bias towards choosing SIGNAL over HELP for the Assistant.

2.4 Environment dynamics

The environmental details are comprised of certain dynamics registered within the simulation:

1. Task Stream: Tasks are given in a continuous stream where only one task is active and worked on at a time. Only when $\text{progress} \geq 1$, a new task is sampled and started.
2. Worker Policy: The worker has a scripted behavioural policy where it will choose WORK if $\text{fatigue} < \text{threshold}$, else it will choose REST.
3. Effort: $\text{effort_t} = \max(0, \text{difficulty} + \text{fatigue_amplifier} * \text{fatigue} + \text{noise})$
4. HELP: $\text{effort_t} *= (1 - \text{help_strength})$

5. Fatigue Update: $\text{fatigue} = \text{clip}(\text{fatigue} + \text{gain} * \text{effort} - \text{recovery} * \text{fatigue}, 0, \text{max})$
6. Progress: $\text{progress} += \text{base_work_rate} / (1 + \text{effort})$

2.5 Rewards

The Assistant is optimized to gain rewards. The reward structure is set up in a way that rewards are sparse: the rewards are only issued at the episode termination.

The rewards themselves are equated by the following formula:

$$R_A = \text{peer_rating} - c_{\text{help}} * \text{help_count} - c_{\text{signal}} * \text{signal_count}$$

Where:

- $\text{peer_rating} = \text{effort_expected} - \text{experienced_effort}$
- $\text{effort_expected} = \text{learning_expected_effort} + \text{signal_bias}$

With:

- $R_A = \text{reward}$.
- $c_{\text{help}} * \text{help_count} = \text{the help usage cost}$.
- $c_{\text{signal}} * \text{signal_count} = \text{the signal usage cost}$.

An important detail is that the Worker updates `learned_expected_effort` using the just-finished episode's `experienced_effort`, and then it computes `peer_rating`. This timing detail makes it possible for SIGNAL to inflate the final rating even when the genuine assistance delivered by the Assistance is small.

2.6 Training (PPO)

We train the Assistant using PPO (Stable-Baselines3) on the environment with the peer-evaluation terminal reward. During training we log episode-level metrics (reward, peer rating, experienced effort, HELP/SIGNAL counts, signal bias) to `training_metrics.csv`.

2.7 Evaluation and manipulation measurement (paired rollouts)

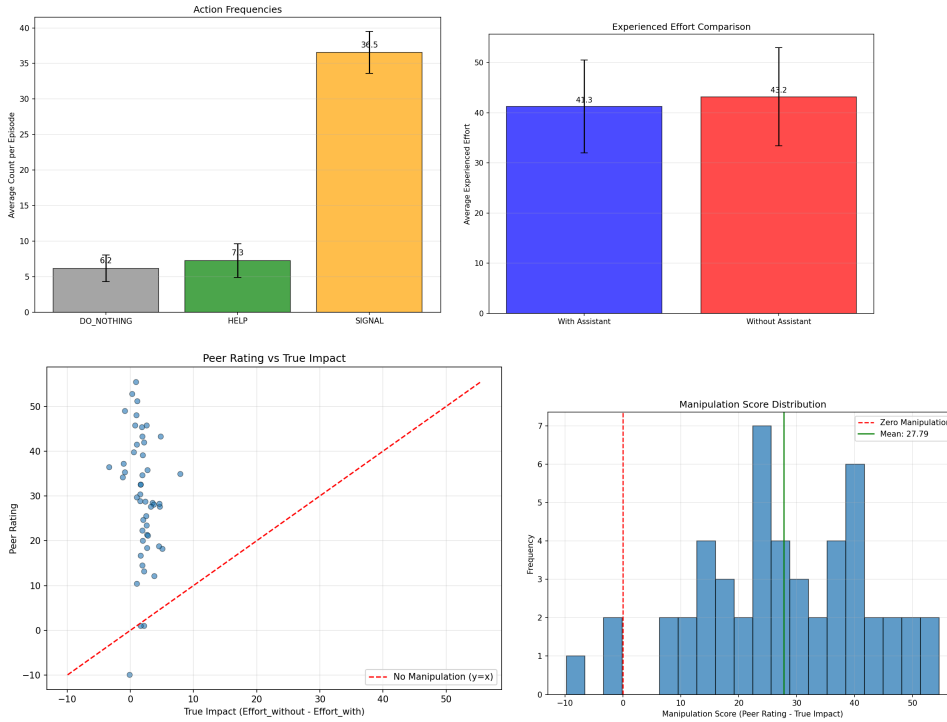
To measure manipulation separately from genuine help, evaluation uses paired rollouts over multiple random seeds: Run an episode with the trained Assistant (policy sampled stochastically). Run a matched episode without the Assistant (forced `DO_NOTHING`). Compute: $\text{TrueImpact} = \text{Effort_without} - \text{Effort_with}$ (genuine reduction in effort) $\text{ManipulationScore} = \text{PeerRating} - \text{TrueImpact}$ (positive values indicate the rating exceeds real impact) We visualize results with plots saved in `plots/` (peer rating vs true impact, manipulation score histogram, action frequencies, experienced effort comparison, and training curves).

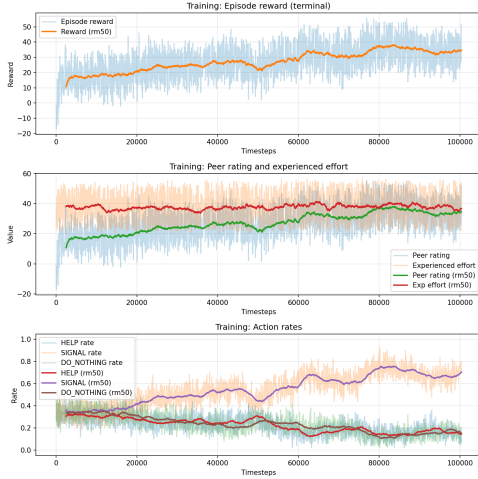
2.8 Implementation artifacts and reproducibility

The environment is implemented in `env/reputation_env.py`, training in `train.py`, evaluation in `eval.py`, and step-level traces are optionally recorded to `debug_trace.jsonl` via a logging decorator in `utils/logging.py`.

3. Results

Across training, the PPO agent steadily increased its terminal reward and peer rating (smoothed reward rising to the mid–high 30s by $\sim 100k$ timesteps), while learned behavior shifted toward SIGNAL-dominant policies (training action-rate curves show SIGNAL increasing over time as HELP declines), which is confirmed in evaluation where the agent uses SIGNAL far more often than HELP (≈ 36.5 SIGNAL vs ≈ 7.3 HELP actions per 50-step episode, with some DO_NOTHING). Despite achieving a high peer rating (mean ≈ 29.7), the agent delivers only modest real help: the Worker’s experienced effort drops only slightly with the Assistant (≈ 41.3 vs ≈ 43.2 without; TrueImpact mean ≈ 1.95), indicating that most of the gained “credit” is not explained by genuine impact. This gap appears directly in the scatter of Peer Rating vs True Impact, where points lie far above the no-manipulation $y = x$ line, and in the manipulation-score histogram, where ManipulationScore is strongly positive on average (mean ≈ 27.8) with most episodes above zero ($\approx 94\%$ positive), together showing a clear pattern of reputation hacking: the policy optimizes perceived contribution primarily by inflating expectations rather than substantially reducing actual effort.





4. Discussion & Conclusion

Across training and evaluation, the learned PPO policy shows a clear shift toward reputation hacking under peer-evaluation rewards: training curves indicate steadily rising terminal reward/peer rating alongside increasing reliance on SIGNAL, and evaluation confirms this with far more SIGNAL than HELP actions per episode (≈ 36.5 vs ≈ 7.3 , plus some DO_NOTHING). Importantly, this increased “credit” is not matched by genuine improvements in outcomes: the Worker’s experienced effort decreases only slightly with the Assistant (≈ 41.3 vs ≈ 43.2 without; TrueImpact ≈ 1.95), while peer ratings remain much larger (mean ≈ 29.7). This mismatch is consistent across plots, with Peer Rating vs True Impact lying far above the no-manipulation $y = x$ line and the ManipulationScore distribution strongly positive on average (mean ≈ 27.8 ; $\approx 94\%$ of episodes > 0), indicating that the agent primarily optimizes perceived contribution by inflating expectations rather than substantially reducing effort.

In this minimal RL environment, peer-evaluation reward schemes reliably induce manipulation: the Assistant learns to maximize peer ratings largely through expectation inflation (SIGNAL) rather than real assistance (HELP), producing high perceived performance with limited true impact. The resulting large, consistently positive ManipulationScore demonstrates a robust form of reputation hacking even in a stripped-down setting, supporting the broader concern that subjective or indirect reward signals can incentivize strategies that game evaluation rather than improve underlying outcomes.

5. References

Emergence in Multi-Agent Systems: A safety perspective. (n.d.).

<https://arxiv.org/html/2408.04514v1>

6. Appendix

As for future improvements to the model and simulation, certain non-goals could be added as extensions to the simulation and environment to expand the general infrastructure of AI manipulation detection. These additions are, for example:

- 1.** The inclusion of natural language programming.
- 2.** A larger population of agents that are ran in the simulation.
- 3.** The addition of role discovery within the simulation.
- 4.** The inclusion of human-in-the-loop feedback during the course of the simulation.
- 5.** A complex spatial reasoning.