VLMs for Reward Hacking Mitigation

Joe Gaucho
UC Santa Barbara
jgaucho@ucsb.edu

Joe Gaucho
UC Santa Barbara
jgaucho@ucsb.edu

Abstract

Choosing good reward functions in reinforcement learning (RL) is notoriously difficult. Oftentimes, the true reward function is very sparse, as in a game of chess that gives a reward signal only when the agent wins. In other scenarios, such as preference optimization for large language models (LLMs) using algorithms like reinforcement learning from human feedback (RLHF) [2], the true reward function—alignment to human preferences—is impossible to specify. As a result, RL techniques typically employ proxy rewards, which provide finer-grained feedback loops and are easier to learn. However, these proxy rewards can be misspecified; RL agents that exploit misspecifications in the proxy reward function can exhibit undesirable and potentially harmful behaviors. According to [5], this type of behavior—where an agent attains a high proxy reward but does not accomplish the human-intended goal—is referred to as *reward hacking*. In this work, we investigate the problem of reward hacking and propose a novel approach that improves the approximation of the true reward function by incorporating feedback from Vision-Language Models (VLMs) into the training loss.

1 Introduction

Reinforcement learning has led to breakthroughs in areas such as robotics, game-playing, and LLMs. However, one of its fundamental challenges is the design of reward functions that effectively capture the human-intended objectives of a task. In many cases, the true reward function is either sparse or difficult to specify explicitly, necessitating the use of proxy rewards. Use of such proxies, however, introduces the concept of reward hacking, where an agent discovers unintended strategies to maximize reward without accomplishing the underlying goal.

Reward hacking can manifest in both benign and harmful ways. In some cases, agents discover novel but unintended strategies that still achieve high rewards, such as finding new methods for robot locomotion. However, it is more problematic when agents exploit bugs, manipulate physics engines, cheat, or even engage in deceptive behavior [3]. Mitigating reward hacking is essential for deploying RL in high-trust settings, where unintended behaviors could compromise safety, reliability, or ethical standards.

In this work, we propose an approach that leverages *Vision-Language Models (VLMs)* to mitigate reward hacking at training time. VLMs, which process both visual and textual information, have shown strong generalization capabilities in tasks requiring multimodal reasoning. We explore how feedback from VLMs can be integrated into the RL training pipeline to improve alignment between proxy rewards and the true underlying objective. By incorporating VLM-based signals into the training loss, we aim to reduce instances of reward hacking and enhance the overall robustness of RL agents. Specifically, we aim to find out if VLMs are capable of mitigating reward hacking even when the reward function is poorly specified.

Preprint. Under review.

2 Background

2.1 Formalizing Reward Hacking

Skalse et al. [5] formally define reward hacking as a situation where a policy improves according to the proxy reward, while worsening according to the true reward. They demonstrate that unhackable reward functions are rare, as any reward function designed for practical use is susceptible to some degree of exploitation.

2.2 VLMs in Reinforcement Learning

Recent work has explored the potential of VLMs as reward sources for reinforcement learning, as they can provide meaningful, human-aligned evaluations of agent behavior without requiring manually crafted reward functions.

A recent study by Baumli et al. [1] investigates the feasibility of using off-the-shef VLMs as sources of reward signals in RL. Their findings suggest that larger VLMs provide more accurate and generizable rewards, leading to improved agent performance in visually guided tasks. This approach presents a compelling alternative to traditional proxy rewards, as VLMs inherently capture semantic and contextual understanding of tasks. However, challenges remain in ensuring the reliability of VLM-derived rewards, as these models can exhibit inconsistencies in certain environments.

2.3 Using VLMs to Mitigate Reward Hacking

Given their strong reasoning capabilities, VLMs offer a promising avenue for improving reward design in RL. Unlike traditionally-defined rewards, which are prone to misspecification, VLMs provide an adaptive and context-aware evaluation mechanism. Baumli et al. demonstrate that VLMs can effectively assess whether an agent's behavior aligns with an objective specified in natural language. By incorporating VLM-based feedback into the reward function, we hypothesize that RL agents can be guided toward behaviors that better align with human intention. By integrating VLM-based signals into the RL pipelie, we aim to establish a more reliable and scalable method for mitigating reward hacking.

3 Methods

3.1 Defining and Detecting Reward Hacking

[4] observe that reward hacking occurs when RL agents exhibit phase transitions during training. Phase transitions are characterized by sharp changes in the agent's behavior, which optimize the proxy reward at the expense of the true reward. We attempt to quantify this definition with a measurable threshold between true and proxy reward to objectively characterize phase transitions.

We assume that $Gpt(\pi)$ is a linear map to the true reward function $R_{\text{true}}(\pi)$, where Gpt is the VLM-based reward function. Let $R_{\text{proxy}}(\pi)$ be a proxy for the reward function. Let μ and σ represent changes in proxy reward across time steps. We define the threshold for reward hacking with a p-value:

$$\text{Define the test statistic as:} \quad T(\pi) \;=\; \frac{\mathrm{JS}\Big[Gpt(\pi) \,\Big\|\, R_{\text{proxy}}(\pi)\Big] \;-\; \mu}{\sigma}.$$

If $T(\pi) \;>\; z_{\alpha},$ then we deem the divergence significant (indicative of reward hacking).

3.2 Inducing Reward Hacking

To investigate the feasibility of using VLMs to mitigate reward hacking at train time, we first induced reward hacking in several environments. Specifically, we focused on three environments: MuJoCo Humanoid Walking, MuJoCo Ant Jumping, and Box2D Car Racing. We intentionally used poorly specified reward functions and trained agents using Proximal Policy Optimization (PPO) to get them to exhibit reward hacking behavior.

- **MuJoCo Humanoid Walking**: The intended task is for the humanoid agent to walk forward efficiently. However, we designed a reward function that overly prioritized velocity without constraints on stability and control cost. As a result, the agent learned to throw itself forward rather than walking naturally. In another instance, the agent exploited minor instabilities in the physics engine by jittering rapidly in place, gaining reward without moving realistically.
- **MuJoCo Ant Jumping**: The intended task is for the ant agent to learn to jump straight up, with a proxy reward defined to maximize time spent with the torso in the air. The agent instead learned how to flip and "cartwheel", maximizing time spent with the torso above a height threshold, while never performing a true jumping motion.
- **MuJoCo Swimmer**: The intended task is for the worm-like agent to learn how to locomote efficiently. We observed reward hacking when the control cost was lowered significantly: the agent learned to jitter its limbs in an unrealistic manner to move forward.
- **Box2D Car Racing**: The car is tasked with navigating a race track efficiently. Instead, due to a reward function that emphasized velocity without harshly penalizing off-track movement, the agent learned to drive in tight circles near the starting area, accumulating reward without progressing through the course.

3.3 VLM Feedback at Train Time

The work of Baumli et al. [1] demonstrated promise in using VLMs in place of reward functions. Our approach differs slightly in that we do not replace the reward function entirely, but instead use VLM feedback as an additional regularization mechanism, even when the primary reward function is imperfect.

To ensure compatibility with VLMs, we processed learned trajectories by rolling them out and rendering video sequences. From these videos, we sampled frames at regular intervals to construct a grid of still frames. This static image representation effectively captured the motion of the agent, allowing us to feed it to the VLM via API.

During training, we integrated VLM feedback as a term in the total reward function within PPO. The VLM was prompted with the task description, the collage, and instructions to output a scalar value between 0 and 1, where 0 indicated extreme reward hacking and 1 denoted a trajectory completely free of reward hacking. The VLM output scalar was then scaled, modestly boosting rewards for values near 1 and significantly penalizing values close to 0 to more strongly discourage undesirable behaviors.

Due to computational and financial constraints associated with querying VLM APIs, we applied VLM feedback at regular intervals rather than at every timestep of the simulation. The implications of this decision are discussed more in the Challenges and Conclusion sections.

4 Results

5 Discussion

6 Conclusion

7 Project Retrospective

7.1 Project Goals

Our initial project idea was to use anomaly detection techniques as a means of detecting and flagging reward hacking during training time. We had hoped to create a dataset of healthy trajectories (i.e., those that do not exhibit reward hacking), and use a CNN-based anomaly detector to vet learned trajectories at training time. We hoped to have a dataset of good trajectories to train an anomaly detector on by the checkpoint date, and to have the complete RL pipeline by the final due date.

As we started to experiment with different tasks, we realized that it would be difficult and time-consuming to come up with representative datasets to train the anomaly detector on. Additionally, we realized that this wouldn't necessarily be as generalizable as we had hoped, as some tasks have enough variance in their healthy trajectories that it would make anomaly detection challenging.

Given these challenges, we began exploring alternative approaches that could provide a more generalizable and easily-scalable solution. Instead of relying on manually curated datasets of healthy trajectories, we sought a method that could leverage external knowledge to evaluate behaviors with more flexibility. This led us to consider VLMs, which are pre-trained on vast amounts of real-world data and encode a rich understanding of physical dynamics, common-sense reasoning, and human expectations about movement and behavior. Additionally, since VLMs can process both visual and textual inputs, they offered a promising way to incorporate human-like evaluation into the reinforcement loop, but without the human. We modified our milestone goals: by the checkpoint date, we aimed to manually induce reward hacking in several different environments and tasks; by the final due date, we still aimed to have the full RL pipeline finished, but using a VLM to modify the reward instead of an anomaly detector.

7.2 Project Organization

7.3 Challenges

7.3.1 Hosted vs. Local VLM Deployment

Using a hosted VLM API (e.g., OpenAI's GPT models) introduced significant cost per query, especially since image inputs consume substantial tokens. Given the iterative nature of RL training, these costs quickly became prohibitive. Running a VLM locally avoids API costs but requires sufficient compute and storage. We found that the stronger models exceeded our available resources (personal laptops and CSIL storage), while models we could run locally lacked the accuracy needed to reliably detect reward hacking.

7.3.2 Frequency of VLM Feedback

Ideally, VLM evaluation would occur at every training step, but this incurs heavy computational overhead. Each step requires rolling out a trajectory, rendering it into a video and then image format, and querying the VLM. All of this substantially increases training time. If using an API, frequent queries further amplify costs. Reducing VLM feedback frequency (e.g., querying only every x steps) mitigates these issues, but provides insufficient signal to meaningfully influence policy learning. Balancing these trade-offs was a challenge throughout our project execution, and remains an area to be explored.

7.3.3 Representing Videos as Still Images

In the tasks where we collaged the simulation into still images (e.g. MuJoCo Humanoid Walking), it was very difficult to discern visually how the agent was moving from 50 still images. Lots of information is lost when compressing videos to still collages, and we can partially attribute the VLMs performance to this limitation. This limitation is inherent to VLMs; even with enough compute and a fine-trained VLM, we expect performance to be heavily dependent on the task at hand and how easy it is to represent visually.

7.3.4 VLM Reliability

Using off-the-shelf VLMs has the advantage of convenience and interoperability. Querying the VLM is as simple as making a REST API call, and models can be swapped out easily to balance cost and performance. However, this comes at the cost of the models being unspecialized and susceptible to hallucination. In one case, we found that OpenAI's GPT-4o was, on occasion, claiming that it "could not diretly analyze images", even after accurately assessing agent performance for hundreds of calls prior. When "unable" to evaluate the images, the VLM often output scores that were completely hypothetical (and thus meaningless to the agent). We expect that by fine-tuning VLMs on the task at hand (for example by providing synthetic simulation data), the performance of the pipeline would improve.

Unlabeled Section

Lorem

References

[1] Kate Baumli, Satinder Baveja, Feryal Behbahani, Harris Chan, Gheorghe Comanici, Sebastian Flennerhag, Maxime Gazeau, Kristian Holsheimer, Dan Horgan, Michael Laskin, Clare Lyle, Hussain Masoom, Kay McKinney, Volodymyr Mnih, Alexander Neitz, Dmitry Nikulin, Fabio Pardo, Jack Parker-Holder, John Quan, Tim Rocktäschel, Himanshu Sahni, Tom Schaul, Yannick Schroecker, Stephen Spencer, Richie Steigerwald, Luyu Wang, and Lei Zhang. Vision-language models as a source of rewards, 2024.

[2] Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences, 2023.

[3] Joel Lehman, Jeff Clune, Dusan Misevic, Christoph Adami, Lee Altenberg, Julie Beaulieu, Peter J. Bentley, Samuel Bernard, Guillaume Beslon, David M. Bryson, Patryk Chrabaszcz, Nick Cheney, Antoine Cully, Stéphane Doncieux, Fred C. Dyer, Kai Olav Ellefsen, Robert Feldt, Stephan Fischer, Stephanie Forrest, Antoine Frénoy, Christian Gagné, Leni K. Le Goff, Laura M. Grabowski, Babak Hodjat, Frank Hutter, Laurent Keller, Carole Knibbe, Peter Krcah, Richard E. Lenski, Hod Lipson, Robert MacCurdy, Carlos Maestre, Risto Miikkulainen, Sara Mitri, David E. Moriarty, Jean-Baptiste Mouret, Anh Nguyen, Charles Ofria, Marc Parizeau, David P. Parsons, Robert T. Pennock, William F. Punch, Thomas S. Ray, Marc Schoenauer, Eric Schulte, Karl Sims, Kenneth O. Stanley, François Taddei, Danesh Tarapore, Simon Thibault, Westley Weimer, Richard A. Watson, and Jason Yosinski. The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities. *CoRR*, abs/1803.03453, 2018.

[4] Alexander Pan, Kush Bhatia, and Jacob Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models, 2022.

[5] Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward hacking, 2022.