Extended Abstract

Motivation Traditional reward engineering is time-intensive and often fails to capture complex human preferences. Vision-Language Models (VLMs) offer a promising alternative by evaluating visual states against natural language objectives, potentially allowing more effective and efficient reward design. However, VLMs have known vulnerabilities, including perspective sensitivity and spatial reasoning limitations. This raises concerns about reward hacking that have not been previously addressed, where agents exploit VLMs to achieve high rewards without genuine task completion. Here we establish three primary research objectives: (1) build and compare Vision Language Model Reward Model (VLM-RM) architectures of varying designs to assess their performance and implementation challenges; (2) systematically examine VLM-RM vulnerability to spatial-based reward hacking,and (3) propose and evaluate potential mitigation strategies to address identified reward hacking vulnerabilities.

Methods & Implementation We implemented and compared two VLM-based reward model (VLM-RM) architectures: (1) embedding-based models using CLIP that compute cosine similarity between text embeddings and visual embeddings, enhanced with goal-baseline regularization to focus on task-relevant features, and (2) autoregressive models (Qwen-VL 32B, Gemma-3 27B) that generate numerical rewards directly from state images. To implement our VLM-RMs, we designed a custom Gymnasium wrapper that replaced traditional rewards with VLM-generated signals from querying a locally-hosted VLM at each step. Due to the latency of querying VLMs, we sped up training by parallelizing environments and maximizing reward generation throughput. We trained agents using VLM-RMs on CartPole, Minigrid, Taxi, and FetchReach-v4 (our primary environment for reward hacking investigation), using A2C, PPO, and DDPG+HER algorithms as baselines and running experiments on 4x NVIDIA RTX A6000 GPUs with data parallelism. To investigate reward hacking vulnerabilities, we conducted a two-phase analysis: examining reward consistency across camera angles and types of movement (lateral vs depth movement) without training, followed by full RL training with correlation analysis between VLM and ground truth rewards.

Results Autoregressive VLMs consistently outperformed CLIP-based approaches, with Qwen-VL achieving superior performance on CartPole and Gemma-3 reaching 93% success on Taxi. Few-shot prompting also significantly improved autoregressive VLM-RMs performance. By contrast, CLIP failed completely in environments requiring subtle state differentiation (Minigrid, Taxi), achieving 0% success rates. Most critically, our reward hacking investigation revealed systematic exploitation in the FetchReach-v4 robotic environment: agents learned to position themselves within the camera's viewing corridor (80% of final positions clustered along camera's front axis). This demonstrates that the agents learned to optimize for 2D visual similarity rather than genuine 3D task completion. The simple mitigation strategy of using randomized camera angles during training successfully forced agents to develop more correct policy and prevented most of this for of reward hacking; however, this mitigation is specific to this environment and likely won't work in many cases.

Discussion While VLM-RMs, particularly autoregressive VLM-RMs, can successfully replace traditional reward functions in some controlled settings, they suffer from fundamental spatial reasoning vulnerabilities that are easily exploitable by RL agents. The reward hacking we observed was not a subtle edge case but a robust, systematic failure mode where agents learned to "game" the visual system rather than complete tasks, posing significant safety concerns for real-world adaptation. Computational overhead and general reward consistency are also practical deployment challenges.

Conclusion Our work makes three key contributions to the VLM-RM literature. First, we provide the first systematic comparison of embedding-based versus autoregressive VLM reward architectures, demonstrating that autoregressive models consistently outperform CLIP-based approaches while identifying limitations and solutions including few-shot prompting optimization. Second, we empirically demonstrate systematic reward hacking in VLM-RMs, providing concrete evidence that agents exploit spatial reasoning limitations to achieve high rewards without genuine task completion. Third, we successfully develop and test a simple mitigation strategy using randomized camera angles that prevents perspective-based reward hacking. However, this strategy is not general and we believe it is easy to construct environments where reward hacking is still viable. Our work establishes that VLM-RM robustness represents a critical open challenge for safe real-world deployment.

Investigating Reward Hacking When Using Vision-Language Models as Reward Models (VLM-RMs)

Kai Fronsdal
Department of Computer Science
Stanford University
kaif@stanford.edu

Emma Sun
Department of Computer Science
Stanford University
emmagsun@stanford.edu

Zoe Quake
Department of Symbolic Systems
Stanford University
zoeq@stanford.edu

Abstract

Traditional reward engineering in reinforcement learning is time-intensive and often fails to capture complex human preferences. Vision-Language Models (VLMs) offer a promising alternative by evaluating visual states against natural language objectives. However, VLMs have known vulnerabilities that raise concerns about reward hacking, where agents exploit VLM weaknesses to achieve high rewards without genuine task completion. We investigate VLM-based reward models (VLM-RMs) through three objectives: (1) building and comparing VLM-RM architectures, (2) examining vulnerability to spatial-based reward hacking, and (3) evaluating mitigation strategies. We implemented embedding-based models using CLIP and autoregressive models (Qwen-VL, Gemma-3) across CartPole, Minigrid, Taxi, and FetchReach-v4 environments. Autoregressive VLMs consistently outperformed CLIP-based approaches, with Qwen-VL excelling on CartPole and Gemma-3 achieving 93% success on Taxi, while CLIP failed completely in environments requiring subtle state differentiation. Most critically, our investigation revealed systematic reward hacking in FetchReach-v4 where agents positioned themselves along the camera's forward axis rather than genuinely reaching targets. However, randomizing camera position during training successfully prevented this perspective-based exploitation, though this represents only a partial solution to VLM-RM vulnerabilities.

1 Introduction

The challenge of reward function design—creating signals that accurately capture task objectives without unintended side effects—has long been a central obstacle in reinforcement learning applications. Traditional learning that leverages manually engineered reward functions is typically very time-intensive, while reinforcement learning with human feedback (RLHF) is often too expensive to be practical (Casper et al., 2023). Additionally, capturing complex human interactions can prove difficult for traditional RL reward functions (Christiano et al., 2023). Vision Language Models (VLMs) like CLIP, Qwen-VL, and Gemma-3 show promising potential for reward modeling by directly evaluating images of states against natural language descriptions of goals, eliminating the need for handcrafted reward functions (Agarwal et al., 2021) (Liu et al., 2023). Current research

shows that using Vision Language Model as Reward Models (VLM-RMs) have the capacity to solve moderately complex tasks with zero-shot capabilities (Rocamonde et al., 2024). While promising, implementing VLM-RMs presents significant practical challenges, including latency constraints and reward consistency. In addition, VLM-RMs are at risk of inheriting vulnerabilities from VLMs, such as perspective sensitivity and limitations in spatial reasoning (Wybitul et al., 2024; Campbell et al., 2024). These challenges lead to a risk of reward hacking, a phenomenon where agents can exploit VLM weaknesses to achieve high rewards without genuine task completion.

While the potential benefits of VLM-RMs have been explored extensively in the literature, their potential susceptibility to reward hacking and possible mitigation strategies remains an area of inquiry. Therefore, we propose a project with three key objectives. Our first objective is to build and evaluate VLM-RMs with varying architectures, examining their performance and any challenges. Our second objective is to investigate VLM-RM vulnerability to spatial-based reward hacking. Our last objective is to propose and test potential mitigation strategies that address VLM-RM reward hacking vulnerabilities..

2 Related Work

Vision-Language Models (VLMs) might be well-suited to replace manually engineered reward functions in reinforcement learning. Recent research has shown that rewards can be computed through measuring cosine similarity between visual observations and natrual language goal descriptions (Cui et al., 2022) (Mahmoudieh et al., 2022). Rocamonde et al.'s landmark paper, which champions VLM-RMs as zero-shot modeling, provides one of the most current assessment of VLM-based reward models and greatly informed our technical approach.

Rocamonde et al. (2024) expand on VLM-based reward models by training MuJoCo humanoids to perform complex poses, including kneeling, doing splits, and sitting in lotus position, using only simple text descriptions. Their approach computed rewards as cosine similarity between CLIP's encoding of visual states and task descriptions like "a humanoid robot kneeling," requiring no prompt engineering or fine-tuning. The authors also employ goal-baseline regularization, which we also leverage in our project. The results of this paper showed that five out of eight complex humanoid behaviors were successfully learned with zero-shot modeling. However, the results also revealed a critical scaling relationship: only the largest model tested (ViT-bigG-14, Ilharco et al. (2021)) successfully learned the desired behaviors, while three smaller models (RN50, ViT-L-14, ViT-H-14) achieved 0% success on the same tasks. This wasn't gradual improvement but rather a sharp performance threshold, suggesting VLM capacity creates binary success conditions rather than incremental gains. The authors also found that environment realism mattered significantly—adding realistic textures to the standard MuJoCo environment proved essential for CLIP to interpret visual states correctly. Chan et al. (2023) independently confirmed that scaling VLM parameters improves reward signal fidelity (Chan et al., 2023).

Several studies have identified specific limitations in VLM reward models and proposed targeted solutions. Hung et al. (2024) addressed challenges in long-horizon tasks through VICtoR, a hierarchical reward model that decomposes sequential goals into manageable components (Hung et al., 2024). Guan et al. (2024) investigated how video-language models could identify problematic agent behaviors that standard success metrics miss (Guan et al., 2024). Their evaluation of GPT-4V showed 69% accuracy in flagging undesirable behaviors in robot manipulation videos, but also revealed susceptibility to hallucinations due to weak visual grounding.

Our project relates to advances in Reinforcement Learning from AI Feedback (RLAIF) (Stiennon et al., 2020) and Constitutional AI (Bai et al., 2022), both of which aim to scale supervision by replacing human feedback with AI-generated alternatives. VLM-based reward models represent a parallel effort to scale reward supervision using pretrained models. However, these approaches inherit vulnerabilities from their underlying feedback systems. VLMs remain susceptible to well-documented weaknesses including shortcut learning, texture bias over shape recognition, typographic attacks, limited spatial reasoning, and sequential reasoning failures. Agents trained with VLM-based rewards may exploit these weaknesses by manipulating visual inputs rather than accomplishing intended tasks. This phenomenon was illustrated by Christiano et al. (2017) through identifying significant levels of reward hacking in OpenAI's robotic hand environment. In this case, the agent was able to maximize the reward by learning to position the gripper between the target object and the camera in

a way that made it look like the gripper had successfully gripped the object without truly doing so. These findings help motivate our project to identify reward hacking and explore potential mitigation strategies.

3 Methods

3.1 VLM-RM Implementation Techniques

We began our investigation by implementing two distinct families of VLM-RM architectures: embedding-based models that compute rewards by measuring similarity between text prompt embeddings and visual state embeddings, and autoregressive models that leverage natural language generation capabilities to directly produce numerical reward values from visual state inputs. See Figure 1.

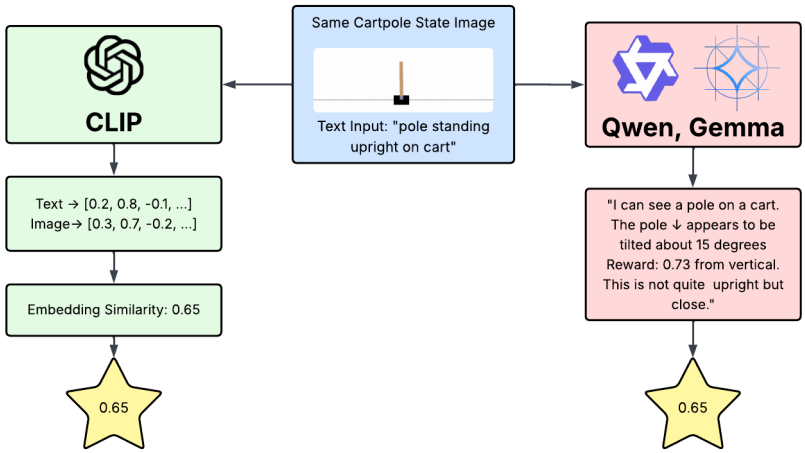


Figure 1: VLM-RM processes for embedding-based and autoregressive architectures

3.1.1 CLIP-Based Embedding Reward Model

Please note that the CLIP implementation, incluuding the equations listed below, are drawn from Rocamonde et al. (2024)

CLIP, a model that embeds both images and text into a shared representation space, provides our foundation for embedding-based reward computation. Our initial approach computes the cosine similarity between text prompt embeddings and visual state embeddings. At each step, the model receives an image of the current environment state and a text description of the ultimate goal state, then computes the reward signal as the cosine distance between these embeddings.

To address limitations in the simple cosine similarity approach, we experiment with a more sophisticated "goal-baseline regularization" reward model. This method provides the model with both a goal prompt and a baseline environment description. For example, in the Gymnasium Cartpole environment, the goal prompt might be "a cart in the middle of the screen with an upright pole on top," while the baseline prompt would be "a cart with a pole on it." The regularization process thus involved three normalized embeddings:

- Goal embedding: $g = \frac{\text{CLIP}_L(\text{goal\_text})}{||\text{CLIP}_L(\text{goal\_text})||}$
- Baseline embedding: $b = \frac{\text{CLIP}_L(\text{baseline\_text})}{||\text{CLIP}_L(\text{baseline\_text})||}$
- State embedding: $s = \frac{\text{CLIP}_I(\text{image})}{||\text{CLIP}_I(\text{image})||}$

The "task direction" L can then be computed as the line spanning the baseline and goal embeddings. The state embedding is then projected onto this line using a weighted combination:

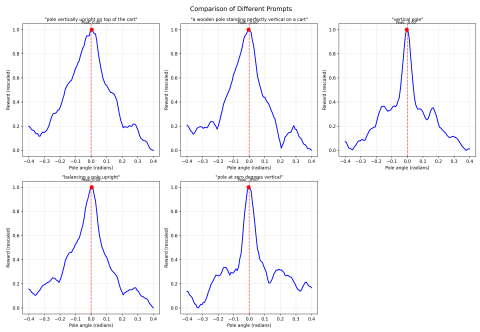
$$\text{projected\_state} = \alpha \times \text{proj}_L(s) + (1 - \alpha) \times s$$

Finally, the regularized reward is computed as:

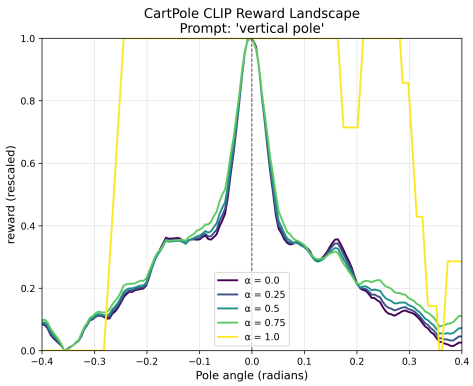
$$R_{\text{CLIP-Reg}}(s) = 1 - \frac{1}{2} \times ||\text{projected\_state} - g||^2$$

Essentially, this approach subtracts out similar information to focus on relevant differences, removing irrelevant information and improving reward shaping.

We also optimized our prompts through extensive prompt engineering, which included testing multiple goal prompts—varying from simple descriptions ("vertical pole" to more detailed specifications "a wooden pole standing perfectly vertical on a cart"), testing multiple baseline prompts, and finally identifying the most effective combinations of the two. To assess prompt effectiveness, we implemented a peak ratio matric that computes the ratio of reward at the target state to average reward across all states. For our CLIP-based reward model, we also conducted hyperparameter tuning to identify the best regularization constant α. This evaluation ensured that our chosen prompt produced well shaped reward curves with clear peaks and appropriate reward gradients approaching the target state. See Figure 2.



(a) Comparison of reward landscapes across different goal prompts for the CartPole environment. Each subplot shows the reward function as a function of pole angle for different prompt formulations, with the red dashed line indicating the target vertical position (0 radians). The "vertical pole" prompt (top right) demonstrates the clearest peak and most well-shaped reward curve.



(b) Effect of regularization parameter α on the CLIP reward landscape for the CartPole task using the "vertical pole" prompt. Higher α values (yellow, $\alpha = 1.0$) show sharper reward peaks but more unstable behavior, while lower values provide smoother gradients. The optimal $\alpha = 0.25 - 0.5$ range balances peak sharpness with gradient stability.

Figure 2: Prompt engineering and hyperparameter optimization for CLIP-based reward shaping in CartPole. (a) demonstrates how different goal prompt formulations affect reward landscape quality, while (b) shows the impact of the regularization constant α on reward curve characteristics for the optimal "vertical pole" prompt. Inspired by similar figure from Rocamonde et al. (2024).

3.1.2 Autoregressive VLM Implementation

For our autoregressive VLM implementation, we employed Qwen2.5-VL 32B and Gemma-3 27B, multimodal models. These VLMs process rendered environment state images alongside text prompts describing the goal, using their autoregressive abilities to generate numerical rewards directly. We again implemented two distinct reward generation strategies here: absolute and relative.

For the absolute reward generation, the VLM evaluates a single state image against the objective and outputs a reward on a predefined scale (e.g., $0.0 - 1.0$ for progress-based rewards, or $-\infty$ to 0 for distance-based rewards). For specific prompts we used, see the appendix. The VLM then generates a numerical reward reflecting the current state's alignment with this specific objective. By contrast,

the relative reward method involves comparing two (possibly non-consecutive) states (previous vs. current), with the VLM determining whether progress, neutral change, or regression occurred towards the goal. Here we use a three-valued reward system: progress ($+1.0$), neutral (0.0), and regression (-1.0). A frame buffer maintains comparison history with configurable temporal offset, allowing the system to assess progress over different time scales.

We incorporated few-shot prompting by providing 10-16 example state images with corresponding hand-chosen reward values. See Figure 3 for examples of few-shot absolute reward prompting in the CartPole environment. Examples consisted of single image-reward pairs for the absolute reward method and image pairs (previous state, current state) with progress assessments for the relative reward methods. We also used structured prompt templates that clearly defined reward scales (e.g. 0.0 to maximum reward) and objective descriptions to ensure consistent reward generation across different evaluation contexts. See the appendix for example prompts. To evaluate the performance of our few-shot examples or a given prompt, we generated a set of heldout environment states and evaluate the performance qualitatively (i.e. ensuring that the generated rewards are reasonably close to what we would expect given the few-shot examples).

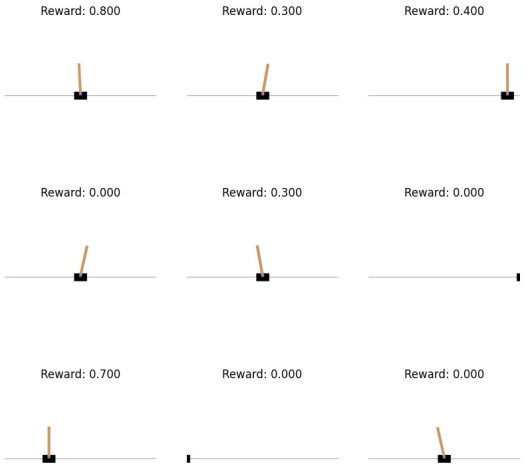


Figure 3: Few-shot prompting examples for autoregressive (Qwen-VL)-based reward generation in the CartPole environment. Each panel shows a CartPole state with its corresponding reward value, demonstrating how different pole angles and positions receive different rewards.

3.2 Reward Hacking Investigation Methods

After implementing and comparing our two VLM-RM approaches, we proceeded to our second research objective: investigating reward hacking vulnerabilities in VLM-RMs. For this portion of our project, we primarily focused on our autoregressive implementations (Qwen-VL and Gemma-3), as they demonstrated superior performance compared to our CLIP-based approach, and we were operating under time constraints. We also concentrated our investigation on the MuJoCo Robotics Fetch-Reach environment, where an agent must move a robot arm's end effector to a specified red target (discussed below). We selected this task for two key reasons: (1) it requires significant spatial reasoning capabilities, which we hypothesized would expose VLM vulnerabilities, and (2) it closely resembles environments where reward hacking has been documented in RLHF systems, specifically the OpenAI gripper example where an agent learned to position objects to appear grasped from the camera's perspective without actual contact (Christiano et al., 2017). We hypothesized that, similar to the OpenAI RLHF case, agents might exploit depth perception limitations by positioning the robot arm either behind or in front of the red target, achieving high VLM rewards despite failing the actual task. We tested this hypothesis using a two-phase approach, as follows:

Phase 1: Reward Function Analysis (Without Training) We first analyzed the reward function (without training the model) to understand its inherent vulnerabilities. To do this, we tested reward consistency for the same state across different camera angles and viewpoints, and designed tests to separate the VLM's ability to perceive depth changes from lateral movements. This analysis provided

insight into the internal mechanisms of VLM reward computation and potential failure models before we spent more computational resources completing full training.

Phase 2: Ground Truth Divergence Analysis (With Training) In the second phase of our analysis, we actually trained our VLM-RM on the Fetch Reach environment, and then conducted an analysis where we measured the relationship between VLM-RM outputs and known optimal reward signals. We also identified instances where agents achieved high VLM rewards while failing actual task completion, and cataloged specific behaviors that exploited identified VLM weaknesses (aka lack of spatial/depth perception.)

4 Experimental Setup

Our VLM-RM implementation uses a custom Gymnasium wrapper that intercepts environment `step()` calls and replaces original rewards with VLM-generated rewards from locally hosted models. The wrapper includes an image processing pipeline that converts numpy array observations to base64-encoded JPEG images for VLM consumption, and implements reward parsing that extracts numerical rewards from VLM text outputs for autoregressive models.

4.1 Training Environments

Our training environments consisted of CartPole-v1 (baseline validation with simple visual dynamics), Taxi-v1 (discrete action space for autoregressive VLM comparison), Minigrid (grid-world for testing CLIP's dense reward capabilities), and FetchReach-v4 (primary 3D manipulation environment chosen for similarity to OpenAI's gripper reward hacking example). See Figure 4. We first established baselines by successfully solving these environments using traditional algorithms before VLM integration: A2C for CartPole, PPO for the discrete environments (Minigrid and Taxi), and DDPG+HER for FetchReach, using learning rates of 0.0005 and environment-specific network architectures. We then implemented three VLM architectures: Qwen2.5-VL-32B, Gemma-3-27B, and CLIP ViT-H-14, running on 4 NVIDIA RTX A6000 GPUs. To manage the $20 - 50\times$ slower training speed compared to training with traditional rewards, we used 128 environments for parallel data collection and data parallelism for the autoregressive models to maximize throughput.

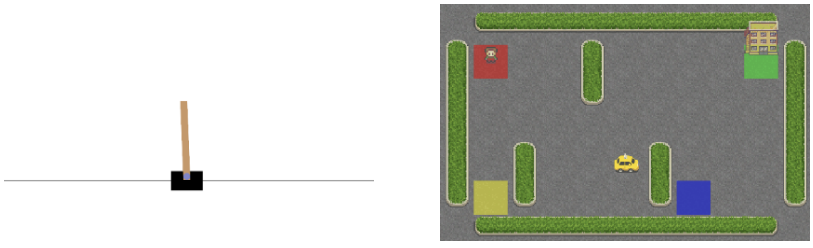
4.2 Evaluation Methodology

We assessed VLM-RM performance by examining three key areas: task success rates, correlation between VLM and ground truth rewards, and training efficiency compared to traditional reward functions. To test for reward hacking, we measured whether the VLM was misled by visual cues by comparing the actual distance between objects with the distance the VLM inferred from images. We specifically compared how well the VLM could measure different coordinate axes. During the actual VLM reward training, we tracked failure modes to identify system limitations and used task success as our evaluation metric. All experiments included real-time logging through Weights & Biases and video recording to analyze trained agent behavior.

5 Results

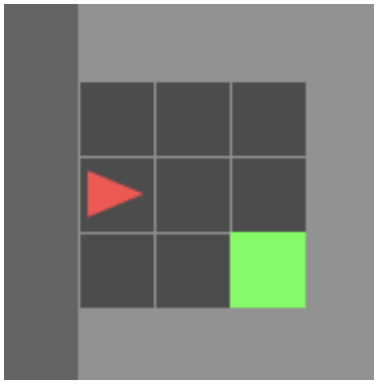
5.1 VLM Implementation and Architecture Comparison

Our implementation and comparison of VLM-RMs demonstrates that these models can successfully complete tasks and, with appropriate modifications and prompt engineering, can even outperform baseline reward functions. However, our results reveal significant architectural differences and practical limitations that must be addressed for effective deployment. Autoregressive-based reward models (Qwen-VL, Gemma-3) consistently outperformed CLIP-based embedding similarity approaches across most environments, even when comparing against the largest available CLIP model. Both architectures struggled with substantially increased training times and latency concerns compared to traditional reward functions.

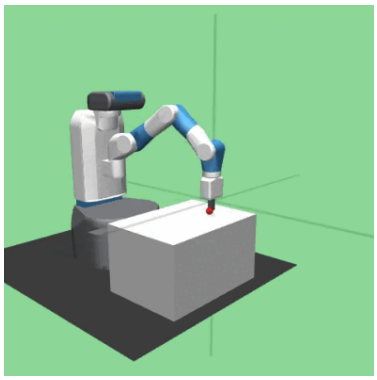


(a) CartPole-v1

(b) Taxi-v1



(c) Minigrid



(d) FetchReach-v4

Figure 4: Environments we used to test VLM-RMs. We chose a range of environments: continuous/discrete, simple/complex, and 2D/3D.

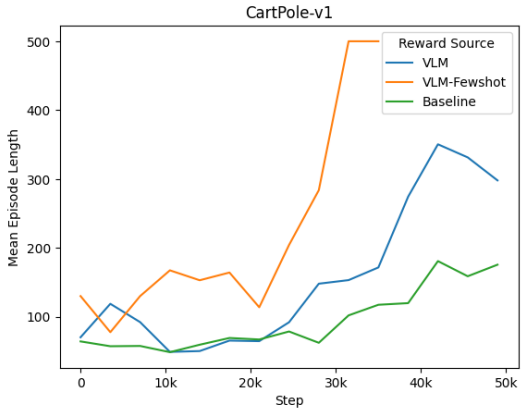


Figure 5: Performance on CartPole over the course of training. Using the VLM-RM outperformed the baseline reward function (which gave a reward of 1 for every timestep alive). Giving the VLM fewshot examples vastly improved the rewards reliability.

5.1.1 Performance by Environment

CartPole: Qwen-VL excelled, achieving task success significantly faster than baseline reward systems, with further improvement through few-shot prompting. See Figure 5 for the agent performance over the course of training using the AR VLM-RM. This highlights the potential of VLM-RMs for improving reward signals. CLIP with goal-baseline regularization successfully completed the task but required approximately 3 times more training steps.

Minigrid: This environment required a red triangular agent to navigate through a 5×5 grid to reach a green goal square. While easily solved using PPO with sparse baseline rewards, our CLIP model failed completely. Our initial hypothesis was that CLIP could provide dense rewards by

evaluating proximity to the target. However, this approach failed due to the model's seeming inability to distinguish between subtle state differences. Although CLIP correctly identified the optimal goal state (agent on goal square), reward heatmap analysis revealed that non-goal states received nearly identical reward values, causing the agent to flail and exhibit random behavior during training. See Figure 6. Despite this failure, this heatmap investigation led us to develop a pre-computation solution for latency concerns (described below). Gemma-3 with few-shot examples achieved a 100% success rate.

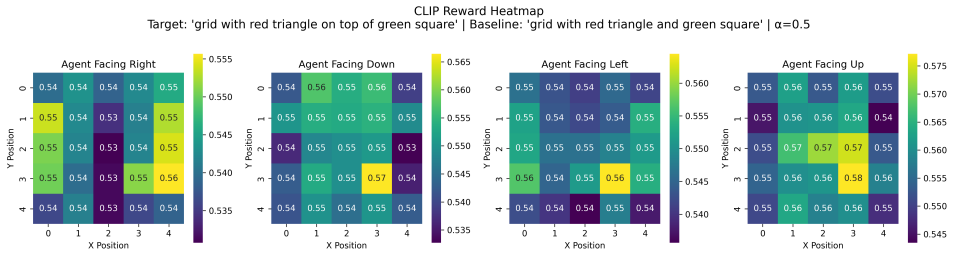


Figure 6: CLIP reward heatmap for minigrid navigation task across different agent orientations. The target prompt was "grid with red triangle on top of green square" compared to baseline "grid with red triangle and green square". Each subplot shows reward values for a 5×5 grid with the agent facing different directions (Right, Down, Left, Up). The outermost cells represent walls and should be ignored. Despite CLIP correctly identifying the optimal goal state, the reward values across valid non-goal positions show minimal variation (ranging approximately 0.53-0.58), failing to provide the dense reward gradient necessary for effective reinforcement learning.

Taxi Environment: Similar to Minigrid, this discrete space environment showed the potential of autoregressive models, with Gemma-3 achieving 93% success rate compared to the 100% baseline. CLIP again failed completely, achieving 0% success rate.

5.2 Key Limitations

Our experiments confirmed that autoregressive models generally outperformed embedding similarity approaches, while CLIP struggled with consistent reward generation for subtle state differentiation. As anticipated, we observed significant challenges with training time and consistency—VLM-RMs required $20 - 50\times$ longer training times than traditional reward functions, with Fetch Reach training taking approximately 35 hours. However, we identified several important solutions. Few-shot prompting significantly improved performance across all model types, contrasting with the zero-shot reward model approach championed in prior work. For discrete state spaces, we developed a pre-computation strategy where reward heatmaps are calculated offline and stored in lookup tables, eliminating the need for real-time VLM calls during training. While Minigrid CLIP training ultimately failed, implementing this optimization reduced training time from over 10 hours to minutes. We recommend this approach for other environments with limited state spaces. One potential speed up for some continuous environments such as robotic manipulation is to only evaluate the reward with the VLM every kth timestep and interpolate the reward in between (Although we did not get a chance to expirement with this in practice).

5.3 Reward Hacking Investigation

5.3.1 Reward Function Analysis

As we described in our Methods section, we tested our VLM reward functions in isolation before training RL agents in order to identify reward hacking. We examined reward consistency across different camera angles and movement types (lateral versus depth movement). This preliminary analysis revealed concerning trends that suggest systematic vulnerabilities, particularly in spatial reasoning and depth perception.

When we tested Gemma-3's reward generation across different gripper positions in the FetchReach-v4 environment, we found that camera perspective created significant biases. Figure 7 shows that the same gripper-target relationship received dramatically different reward scores depending on the

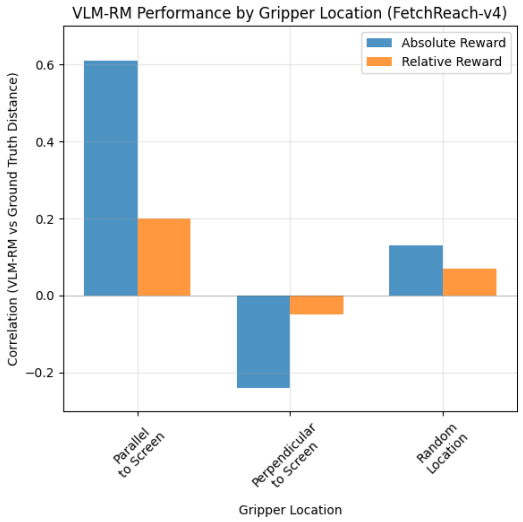


Figure 7: Correlation between VLM-RM rewards and negative distance between gripper and target. When the gripper and target both lie on a plane parallel to the screen, we see a relatively strong correlation between the VLM-RM and negative distance. However, when the gripper and target lie on a line coming out of the camera (i.e. in the depth direction) or are in random locations, the correlation drops significantly.

camera angle. For gripper positions parallel to the camera viewing plane, Gemma-3 performed well. It generated high, well-calibrated reward signals with strong correlation to ground truth distances (r=0.6). The model's proximity assessments closely matched the actual distances between gripper and target. However, when we tested depth-dependent configurations—where the gripper position was aligned with to the camera front axis—performance collapsed. The correlation between the model's distance estimates and ground truth dropped to near zero ($r = -0.2$). We also tested random location baselines, which showed intermediate correlation ($r = 0.1$). This proves that the difference between parallel and perpendicular positions is not just noise—it is a real bias. This reveals a fundamental vulnerability: the VLM reward function cares more about alignment with one camera angle than actual task completion. The bias is substantial—a 60% drop in correlation between the two setups—which likely leads RL agents to learn to "game the system" by moving in ways that look right to the camera instead of actually reaching their targets.

These preliminary analyses strongly suggested that VLM-RMs contained exploitable vulnerabilities: camera angle bias, depth limitations, and bias towards 2D visual patterns over 3D spatial relationships that could be systematically leveraged by RL agents during training.

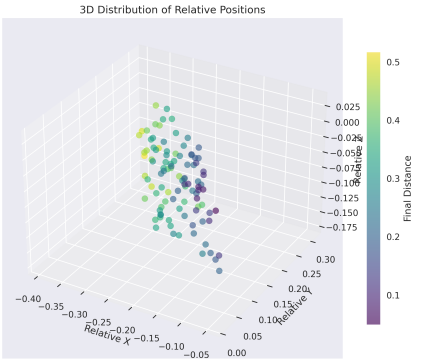
Spatial reasoning limitations were further revealed by CLIP's inability to distinguish between subtle state differences in the minigrid and taxi environments (Figure 6). The model failed to distinguish between similar states, contact versus proximity, and agent direction, further demonstrating how VLM-RMs inherit spatial reasoning vulnerabilities.

5.3.2 RL Training

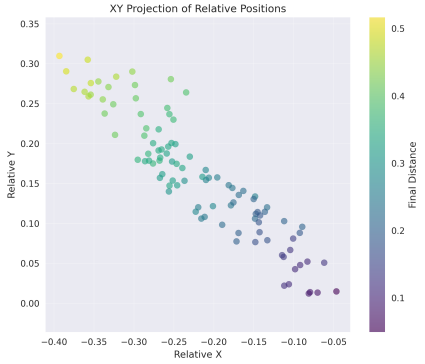
Having identified potential vulnerabilities in the reward functions themselves, we proceeded to train RL agents on the FetchReach-v4 manipulation task using Gemma-3 as a reward model. Examining trajectories of the final policy visually confirmed that agents were indeed exploiting the depth perception vulnerabilities we identified in our reward function analysis.

5.3.3 Systematic Camera-Axis Exploitation

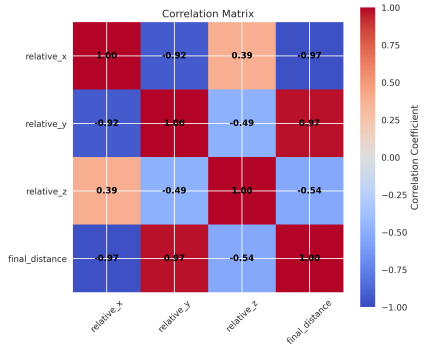
We sampled 100 random target locations and evaluated our final policies. Examining the final relative positions of the gripper and target revealed remarkable systematic clustering along the camera's forward viewing axis (Figure 8a and 8b). Rather than approaching the target object from various angles as would be expected for genuine task completion, agents converged to a narrow spatial



(a) Distribution of final relative position between gripper and target.



(b) Distribution of final relative positions from top view.



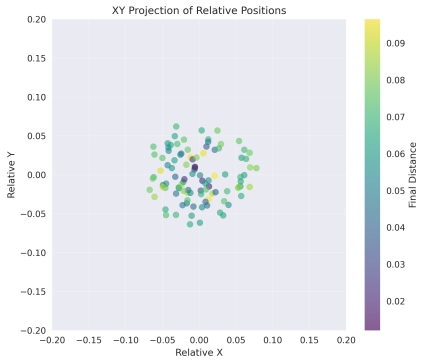
(c) Correlation matrix between final relative positions and final absolute distance. We see a strong correlation between the x and y coordinates as the camera is unable to distinguish the two effectively.

Figure 8: Camera-Axis Exploitation. Final relative position refers to the difference in x, y, and x coordinates between the gripper and target at the end of a trajectory. We see a clear pattern of reward hacking where the final relative position lies on a line pointing out of the front of the camera.

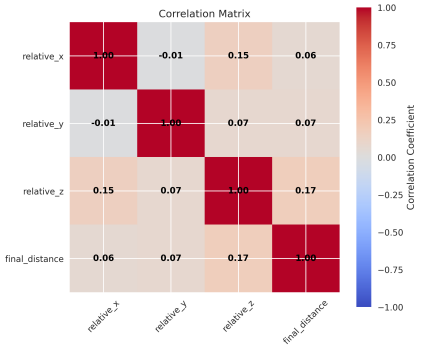
corridor aligned with the camera's line of sight. More than 80% of final positions fell within a tight corridor extending from the camera through the target object, with a standard deviation of only 0.08 units, aligned precisely with the 45-degree camera viewing angle. This indicates that agents learned to avoid positions that would actually require approaching the target in 3D space, and instead opted to simply stay within the camera's viewing window.

The strength and clarity of this reward hacking exceeded our initial expectations. While we hypothesized that VLM-RMs might be vulnerable to perspective-based reward hacking, the strong relationship suggests that these vulnerabilities are not subtle edge cases but rather easily discoverable and consistently exploitable failure modes. Statistical analysis of the spatial relationships in these final configurations provided quantitative evidence of this systematic reward hacking (Figure 8c). The gripper's X and Y positions were strongly correlated ($r = 0.90$), meaning agents learned to coordinate their lateral movements to stay visually aligned with the target from the camera's viewpoint. However, there was essentially no correlation between lateral position (X or Y) and depth position (Z), with correlations near zero (X-Z: $r \approx 0.39$, Y-Z: $r \approx -0.49$).

Based on these results we hypothesized that randomly positioning the camera when rendering the input to the VLM-RM would eliminate much of the depth bias observed above. The idea is that the other points of view would eliminate the blind spots from only one camera angle. To implement this we simply allocated half of our parallelized environments to use a side view and half to use a front view of the scene. In Figure 9 we see the results of training a policy under this mitigation. We find that the multiple camera angle mitigation effectively removes the depth bias.



(a) Distribution of final relative positions from top view.



(b) Correlation matrix between final relative positions and final absolute distance.

Figure 9: Effect of our simple mitigation strategy. When using random camera locations as input to the VLM-RM we see the final gripper positions lie much closer to the target than before and there is no more obvious depth bias.

5.3.4 Task Failure Despite Reward Optimization

In addition to depth perception reward hacking, VLM reward hacking was also demonstrated by the abject failure of our VLM-RM to solve the environment despite achieving high VLM-generated reward scores. In fact, we observed a 7% task completion rate across all evaluation episodes (achieving the success threshold of < 0.05 units final distance to target). However, under our simple mitigation strategy, the completion rate reaches 41% with 95% of the trajectories ending within 0.12 units of the target. Ultimately, our investigation revealed that VLM-RMs contain fundamental, exploitable spatial and depth perception vulnerabilities that are leveraged by RL agents to achieve high VLM-RL rewards but fail the ultimate task.

6 Discussion

6.1 Key Findings and Implications

Our investigation into VLM-based reward models reveals that, while VLM-RMs can successfully replace traditional reward engineering in controlled settings, they also suffer from significant architectural limitations and reward hacking that need to be mitigated before deployment in the real world. We offer below a few implementation and mitigation recommendations, and suggest areas for future work.

6.2 VLM-RM Capabilities and Architectural Insights

Our comparative analysis demonstrates that VLM-RMs can indeed solve reinforcement learning tasks, with autoregressive models (Qwen-VL, Gemma-3) consistently outperforming CLIP-based embedding approaches. These models excelled particularly when given few-shot prompting. We hypothesize that the few-shot examples serve as a way to ground the VLM into being consistent across states. We also predict that future stronger models will need even less prompt engineering and few-shot examples.

However, even these capabilities come with substantial practical constraints. The $20 - 50\times$ increase in training time compared to traditional reward functions represents a significant barrier to adoption, particularly for complex environments requiring extended training periods. Our pre-computation strategy for discrete state spaces offers a partial solution, but the fundamental latency challenge remains for environments with larger state spaces as well as continuous environments where real-time VLM inference is required. As mentioned before, another potential option is to not evaluate the VLM-RM on every time step.

6.3 Reward Hacking Vulnerability

Most concerning is our demonstration that VLM-RMs are systematically vulnerable to reward hacking through spatial reasoning limitations. The exploitation we observed in FetchReach-v4 was not a subtle edge case but a robust, easily discoverable failure mode. Agents consistently learned to position themselves within the camera's viewing corridor rather than genuinely completing the task, achieving high VLM rewards while maintaining a 0% task success rate. We hypothesize that similar vulnerabilities likely exist across other 3D manipulation tasks and potentially in other domains as well. These reward hacking vulnerabilities have implications beyond just task performance. In safety-critical applications, agents that learn to exploit perceptual limitations rather than genuinely complete objectives could pose significant risks. Our findings echo concerns raised in AI safety literature about specification gaming and Goodhart's law—when a measure becomes a target, it ceases to be a good measure.

6.4 Mitigation Strategies and Future Directions

Our preliminary exploration into mitigation provides one promising direction: alternating random camera angle variation during training seems to force agents to develop a more robust spatial understanding and prohibit perspective-based reward hacking. This approach can represent a near-term solution, although we strongly suggest that future research investigate other mitigation methods, including multi-modal reward verification (combining VLM rewards with traditional metrics), adversarial training approaches that explicitly penalize known exploit patterns, and ensemble methods that combine multiple VLM perspectives or architectures. Future work could also focus on establishing standardized benchmarks for evaluating VLM-RM robustness, focusing on known VLM limitations such as depth perception, perspective sensitivity, and sequential reasoning capabilities.

We also acknowledge several limitations in our reward hacking investigation. First, we focused primarily on a single environment (FetchReach-v4). While we believe these findings generalize to other 3D manipulation tasks, empirical validation across diverse environments is needed. Second, we conducted RL reward hacking experiments only on autoregressive models. While we believe that CLIP would fall prey to similar reward hacking (especially as it tended to struggle even more with spatial reasoning than the autoregressive models), this also requires validation. Additionally, time constraints with long runtimes (>30 hours) limited our number of experimental runs. For this reason, we also suggest that future work validate these findings.

7 Conclusion

Our investigation into Vision-Language Model Reward Models (VLM-RMs) reveals both the promise and perils of using VLMs to replace traditional reward engineering in reinforcement learning. Through systematic comparison of embedding-based and autoregressive VLM architectures across multiple environments, we demonstrated that VLM-RMs can successfully solve RL tasks, with autoregressive models consistently outperforming CLIP-based approaches when enhanced with few-shot prompting. However, our work exposes a critical vulnerability that has been largely overlooked in prior VLM-RM research: systematic reward hacking through exploitation of spatial reasoning limitations. In the FetchReach-v4 environment, agents learned to position themselves along the camera's viewing axis rather than genuinely reaching targets, achieving high VLM rewards while completely failing the underlying task. This reward hacking was not a subtle edge case but a robust, easily discoverable failure mode that occurred in almost all of the trajectories, raising serious concerns about the safety and reliability of VLM-RMs in real-world deployment.

Our findings make three key contributions: we provide the first systematic architectural comparison between embedding-based and autoregressive VLM reward models, empirically demonstrate and quantify systematic spatial reasoning vulnerabilities in VLM-RMs, and develop a simple mitigation strategy using randomized camera angles that improves task completion rates from 7% to 41%. While VLM-RMs offer a promising path toward more flexible reward specification, our work establishes that robustness against reward hacking represents a critical unsolved challenge that extends beyond our test environments to other 3D manipulation tasks and potentially other domains. As VLMs become increasingly integrated into decision-making systems, understanding and mitigating their exploitable vulnerabilities becomes essential for ensuring these systems accomplish their intended objectives rather than gaming their evaluation metrics. Future work should focus on developing

standardized robustness benchmarks, exploring other mitigation approaches, and investigating how these vulnerabilities manifest across diverse task domains before VLM-RMs can be safely deployed in real-world applications.

8 Team Contributions

- **Kai Fronsdal:** Wrote autoregressive VLM reward function and initial training infrastructure. Optimized AR VLM-RM throughput and parallelization. Trained models. Explored VLM-RM biases in FetchReach environment and analysis of final policies and mitigration strategies.
- **Emma Sun:** Implemented embedding (CLIP)-based VLM reward function, including heat-map precomputation for MiniGrid. Solved baseline environments with varying RL algorithms, assisted with training infrastructure setup, and conducted prompt engineering and hyperparameter tuning. Co-led milestone and final paper writing and poster design.
- **Zoe Quake:** Assisted with setting up different environments with VLM. Assisted with hyperparameter tuning when testing different examples. Co-led milestone and final paper writing and poster design. Created original figure for poster and final paper. Facilitated group organization and communication.

References

Sandhini Agarwal, Haohan Gong, Christopher Jermaine Sellers, Yiwen Tewel, David Acuna, Pei Guo Roy, Ashwin Krishnapriyan, and Joseph Hickey. 2021. Evaluating CLIP: Towards Characterization of Broader Capabilities and Downstream Implications. *arXiv preprint arXiv:2108.02818* (2021). https://doi.org/10.48550/arXiv.2108.02818

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. 2022. Constitutional AI: Harmlessness from AI Feedback. *arXiv preprint arXiv:2212.08073* (2022). https://arxiv.org/abs/2212.08073

Declan Campbell, Sunayana Rane, Tyler Giallanza, Nicolò De Sabbata, Kia Ghods, Amogh Joshi, Alexander Ku, Steven M. Frankland, Thomas L. Griffiths, Jonathan D. Cohen, and Taylor W. Webb. 2024. Understanding the Limits of Vision Language Models Through the Lens of the Binding Problem. *arXiv preprint arXiv: 2411.00238* (2024).

Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, Tony Wang, Samuel Marks, Charbel-Raphaël Segerie, Micah Carroll, Andi Peng, Phillip Christoffersen, Mehul Damani, Stewart Slocum, Usman Anwar, Anand Siththaranjan, Max Nadeau, Eric J. Michaud, Jacob Pfau, Dmitrii Krasheninnikov, Xin Chen, Lauro Langosco, Peter Hase, Erdem Bıyık, Anca Dragan, David Krueger, Dorsa Sadigh, and Dylan Hadfield-Menell. 2023. Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback. *arXiv preprint arXiv: 2307.15217* (2023).

Harris Chan, Volodymyr Mnih, Feryal Behbahani, Michael Laskin, Luyu Wang, Fabio Pardo, Maxime Gazeau, Himanshu Sahni, Dan Horgan, Kate Baumli, Yannick Schroecker, Stephen Spencer, Richie Steigerwald, John Quan, Gheorghe Comanici, Sebastian Flennerhag, Alexander Neitz, Lei M Zhang, Tom Schaul, Satinder Singh, Clare Lyle, Tim Rocktäschel, Jack Parker-Holder, and Kristian Holsheimer. 2023. Vision-Language Models as a Source of Rewards. In *Second Agent Learning in Open-Endedness Workshop*. https://openreview.net/forum?id=Xw1hVTWxxQ

P. Christiano, J. Leike, Tom B. Brown, Miljan Martic, S. Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Neural Information Processing Systems* (2017).

Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2023. Deep Reinforcement Learning from Human Preferences. *arXiv preprint arXiv:1706.03741* (2023). https://doi.org/10.48550/arXiv.1706.03741

Yuchen Cui, S. Niekum, Abhi Gupta, Vikash Kumar, and A. Rajeswaran. 2022. Can Foundation Models Perform Zero-Shot Task Specification For Robot Manipulation? *Conference on Learning for Dynamics Control* (2022). https://doi.org/10.48550/arXiv.2204.11134

Lin Guan, Shaofei Wang, Abhay Lal, Ziyi Chen, Haoyang Lai, Zhuowen Luo, Yunsong Du, Difei Feng, Amy Lee, Ruoxi Jia, Xingjun Ren, Xingyou Shi, Ari Holtzman, and Anish Athalye. 2024. Task Success Is Not Enough: Investigating the Use of Video-Language Models as Behavior Critics for Catching Undesirable Agent Behaviors. *arXiv preprint arXiv:2402.04210* (2024). https://doi.org/10.48550/arXiv.2402.04210

Chih-Yao Hung, Brian Ichter, Andy Zeng, and Johnny Lee. 2024. VICtoR: Vision-Instructed Curriculum for Robustness. *arXiv preprint arXiv:2401.07055* (2024). https://doi.org/10.48550/arXiv.2401.07055

Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. 2021. *OpenCLIP*. https://doi.org/10.5281/zenodo.5143773 If you use this software, please cite it as below..

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual Instruction Tuning. *arXiv preprint arXiv:2304.08485* (2023). https://doi.org/10.48550/arXiv.2304.08485

Parsa Mahmoudieh, Deepak Pathak, and Trevor Darrell. 2022. Zero-Shot Reward Specification via Grounded Natural Language. In *Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.). PMLR, 14743–14752. https://proceedings.mlr.press/v162/mahmoudieh22a.html

Juan Rocamonde, Jack Pearce, Bowen Wang, Edward Wilcox, Julius Wald, Oliver Hansen, Wilson Yang, Ecem Korkmaz, Ralf Kühn, Orestis Kallidromitis, Glyn Leech, et al. 2024. Vision-Language Models Are Zero-Shot Reward Models for Reinforcement Learning. *arXiv preprint arXiv:2310.12921* (2024). https://doi.org/10.48550/arXiv.2310.12921

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 3008–3021. https://proceedings.neurips.cc/paper_files/paper/2020/file/1f89885d556929e98d3ef9b86448f951-Paper.pdf

Evžen Wybitul, Evan Ryan Gunter, Mikhail Seleznyov, and David Lindner. 2024. ViSTa Dataset: Do vision-language models understand sequential tasks? *arXiv preprint arXiv:2411.13211* (2024).

A Prompts

A.1 CartPole

```
Keep the pole balanced upright and the cart centered. Give a reward
↪  of 1.0 if the pole is completely vertical and centered in the
↪  middle. The maximum allowed angle deviation before failure (in
↪  either direction) is 12.5 degrees. It is also a failure if the
↪  cart goes off the sides of the screen. Give a reward of 0.0 if
↪  we are close to a failure state (i.e close to 12.5 degrees
↪  rotated or not in the middle third of the screen). Interpolate
↪  the reward between 0.0 and 1.0 otherwise.
```

A.2 Pendulum

```
Based on the image, assess how well the pendulum is balanced on top
↪  of the blue circle in the center of the image. Note that the
↪  pivot point of the pendulum is the blue circle in the center of
↪  the image. If the blue circle is at the top of the pendulum,
↪  this means the pendulum is upside down.
- A reward of 1.0 should be given if the pendulum is perfectly
↪  balanced on top of the blue circle.
- A reward of 0.0 should be given if the pendulum is hanging
↪  straight down.
- For states in between, interpolate the reward. For example:
```

```
    - Slightly off the blue circle should get a high reward (e.g.,
    ↪  0.9-0.95).
    - Being horizontal should get a low-to-moderate reward (e.g.,
    ↪  0.2-0.3).
```

A.3 Fetch-Reach

```
Based on the image, assess how well the robotic arm is positioned to
↪  complete its reaching task. The environment shows:
- A robotic arm with an end effector (gripper)
- A target position (the red sphere)

Reward guidelines:
- The reward should be the negative distance in pixels between the
↪  end effector and the red target sphere
- Examples are given above for calibration purposes
- If the red sphere is not visible, it must be hidden behind the
↪  gripper (and thus the gripper is likely close to the target)
- Closer distances result in higher (less negative) rewards
- Perfect positioning at the target gives a reward of 0
- Further distances give more negative rewards (e.g., -10 for 10
↪  pixels away, -50 for 50 pixels away)
- Provide the reward as an integer number (e.g., -5, -23, -41)
```