REWARD MODELING FROM GPT4-VISION PREFERENCES

Bryce Wong, Jonathan Lu & Laryn Qi *†
Electrical Engineering and Computer Science
University of California, Berkeley

1 EXTENDED ABSTRACT

Previously, researchers have determined that a human arbitrator can effectively train a reward model for a reinforcement learning (RL) agent, as opposed to hand-engineering a reward for a task. With the recent rise of vision-language models (VLMs), we attempt to replace the human arbitrator with OpenAI's GPT-4 VLM, as the VLM is much cheaper than using human labeling and we can benefit from improvements in foundational models without much tinkering.

Our reward modeling setup is very similar to (Christiano et al., 2023)'s original structure, although we conduct extensive experimentation with prompting and visual cues to aid the VLM. We use a mixture of SAC and PPO for the policy training, where SAC is used initially to improve agent exploration during segment collection and PPO is used to optimize the final performance. The reward model consists of an ensemble of simple MLP networks, and the model computes the reward of a state by passing the state's observation and action tensors into each MLP network and calculating the average of each network's output. More specific details can be found in the Design section.

We collect segments of agent trajectories to be rated by randomly sampling at each iteration through the preference training loop. Then, an arbitrator (either human or VLM) compares pairs of segments and selects the segment whose visual best satisfies the target task. We utilize these preferences to train the reward model to assign higher rewards to segments that the arbitrator prefers.

We conducted experiments on a variety of Gymnasium environments (Brockman et al., 2016) with varying degrees of success. For each environment, we used the results of human preference training as the baseline to compare with VLM performance. First, we examined if VLM preference training could solve CartPole. We tested a variety of prompts and visual cues in the image to guide the model; however, we found that GPT4-Vision was unable to perform the task at all. We then tested on LunarLander, to see if the model would be able to place the spaceship in the upper left corner of the frame, which also did not work. Additionally, we also tried a few objectives with Walker2D to see if the model could make the agent squat or do the splits. Ultimately, we found the splits task to be most successful with the final agent achieving the splits goal around 60-70% of the time.

Figure 1: Walker2d trained to do splits using GPT4-V preferences. Video performance found here.

*{brywong,jonathan.lu,larynqi}@berkeley.edu
†Names in alphabetical order

2 INTRODUCTION

Within the past few years, significant progress has been made in the development of VLM's, or vision-language models. VLMs are unique in their ability to process both visual and text-based inputs, allowing them to perform a much wider range of tasks. For example, a VLM can be trained to generate captions for different input images, or answer questions about the contents of a visual source (Alayrac et al., 2022). They've also been used as success detectors for specified tasks (Du et al., 2023). This recent development motivates the aim of our project: determining whether a reinforcement learning (RL) agent can leverage the world knowledge of pretrained VLMs to achieve complex tasks.

Previous work has explored whether human preferences can be leveraged to effectively train a reward model for an RL agent (Christiano et al., 2023). Although these methods are effective, they require a lot of human interaction to train the RL agent, making this type of strategy extremely expensive and time-consuming to operate. Our approach is to replace the "human" component of this training process with a VLM. As opposed to prior work, our method aims to use off-the-shelf, black box models as currently, the most powerful VLMs are arguably proprietary, closed source models. Any improvements to the base model can directly be incorporated into our system for better downstream performance.

3 DESIGN

3.1 OVERVIEW

Our objective is to train an RL agent on a task with no predefined reward function using VLM feedback. Our approach is two-fold: first, we use the world knowledge of VLMs to train a reward model for a target task. The reward model will take in the observation and action from RL agent rollouts and will be trained to assign higher rewards to observations that the VLM prefers. Then, we use this reward model to directly train an agent using a state-of-the-art RL algorithm. Our training loop is as follows:

1. The agent's current policy π interacts with the environment and collects visual sources (images or videos of its current observation) in addition to other raw information (observation/action tensors). The policy π's parameters are updated using an RL algorithm from Stable-Baselines3 to maximize the current reward function.
2. We create pairs of segments (s_1, s_2) from the trajectories collected from the previous step.
3. An external arbitrator compares pairs of collected segments and decides which segment is closer to achieving the target goal.
4. The reward model is trained to fit the preferences specified by the arbitrator's decisions.
5. Repeat

Here, we will utilize two different arbitrators for this task: a baseline human arbitrator and OpenAI's GPT-4 vision model (gpt-4-vision-preview). Also, although our approach supports segments of arbitrary length, we evaluate on length one segments (i.e. images) for all our experiments to simplify the task.

3.2 HANDLING PREFERENCES

In order to collect preferences throughout the entire training cycle, we implement a multiprocessing architecture for our system. Namely, we have two processes, one to collect Segments and handle the main training loop of the policy and reward models, and another to receive Segments and collect preference ratings from a human or AI. We created a Preference Database that stores each snapshot along a trajectory as a Segment object, where each Segment stores the visual frame(s) of the environment, the observations, actions, and rewards that correspond to each frame. The database only stores the last 100 Segments provided by the model to ensure that preferences are being chosen from up-to-date data.

As Segments are added to the database, we use a custom Preference Interface (See Figure 2 and Figure 3) to elicit feedback and generate data for the reward model to train on. At each iteration, we sample two Segments (s_1, s_2) at random from the Preference Database and have the arbitrator (human / VLM) decide which visual source appears closer to achieving the current task. If the arbitrator chooses s_1, we assign that pair a probability distribution of $[1, 0]$; otherwise, we assign that pair a probability distribution of $[0, 1]$. We also support a NEITHER choice, where the probability distribution is uniform over the two; however, we found that this doesn't work with GPT4-V sometimes since the model will often hedge and output NEITHER unnecessarily.

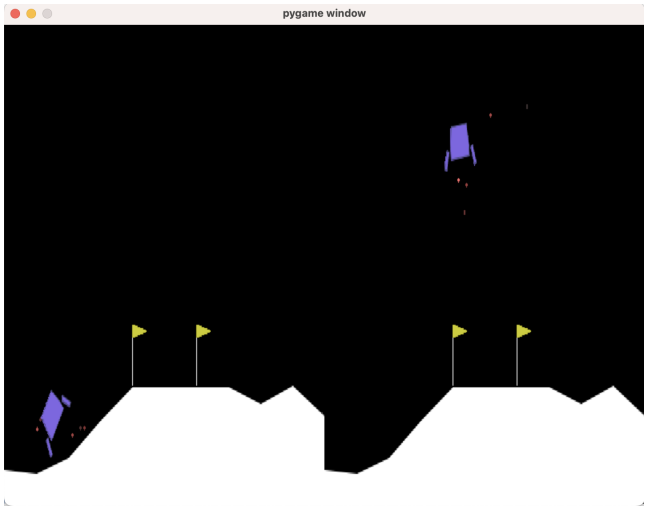

Figure 2: Human-Labeling Preference Interface
The human arbitrator selects their preferred segment using a corresponding hotkey (LEFT, RIGHT, NEITHER). After selection, the interface would refresh with a new pair of Segments.

3.3 Policy Training

We tested a few RL algorithms to train the policy, most notably PPO (Schulman et al., 2017) and SAC (Haarnoja et al., 2018). During the main training loop, we use SAC as it also contains a max entropy objective. We believe that more random policies would be best for the segment collection step to ensure ample coverage from the preference labels of important states. At each epoch, the current policy π first explores the environment for $num\_steps\_explore$ steps before starting the first training rollout. This process allows the agent to populate the Preference Database with Segments to train the reward model with.

Since calibrating the best proportions of policy training steps and reward training steps per loop of the training algorithm is difficult, we added an additional training stage for the agent at the end. We take a static reward model and run a standard RL algorithm like PPO to maximize performance. Although prior work mentioned this can lead to reward hacking (Skalse et al., 2022), we found this was not a particularly big issue for our experiments.

3.4 Reward Model

The custom reward model is designed as an ensemble of MLP networks, where each MLP consists of five linear layers with LeakyReLUs (Xu et al., 2015) as activations. We designed the reward model to either take in just the observations or the observations and actions and didn't find much of a difference. However, we mainly used it with just the observations as we only used images for preference ratings instead of videos so the action isn't visible to the human rater anyway.

To calculate the reward, a given observation o and action a is concatenated into one tensor t or just o if we don't include actions and pass t into each network in the ensemble. The reward is calculated as the mean of the network outputs.

We train the model on the probability distributions generated by the arbitrator between pairs of segments s_1 and s_2, where a higher probability for s_1 indicates that the observation and action in s_1 is closer to achieving the target task (and vice versa). To calculate this probability distribution, we have the model generate a sum of rewards over all the frames for each segment, then take a softmax over the two values to generate the probability distribution. We then use cross-entropy loss, treating the arbitrator's distribution as the correct label, and backpropogate the loss through the ensemble.

3.5 Environment

In order to utilize our reward model in place of the base rewards of Gymnasium environments, we create a custom environment that replaces the default reward function with our reward model. We also change some environment configurations to better enable the agent to perform the tasks we want it to do, like changing the termination criteria or joint gains for movement. Specifically, we tested our system on environments like CartPole, LunarLander, Swimmer, HalfCheetah, and Walker2d but record the ones with interesting findings.

3.6 GPT4-V Integration

We tested a few approaches to query GPT4-V for visual preferences. The list of prompts we used for the tasks and environments is listed in the appendix. In order to save on token usage and to retain consistency between the human and VLM preference visuals, we first tried querying the model by combining two images side by side. We found that the model strongly biased a LEFT output, to the point of being practically unusable. We tried reducing this bias by sending two queries to GPT4-V, with the second having swapped the order of the segments in the image (i.e. (s1, s2) and (s2, s1)). We then only recorded the preference if the model selected the same segment, regardless of whether it was positioned on the right or left. Although this approach seemed to filter out bad responses, it was extremely inefficient and expensive due to the overwhelming proportion of bad responses, so we abandoned the approach.

We additionally tried sending the images as two different messages and asking the model to choose between the first message or the second message. This still did not improve the bias as the VLM instead chose to output FIRST the vast majority of the time.

We considered using video inputs as well since GPT4-V technically supports them; however, we were concerned about token usage and didn't believe the VLM would be able to reason about them any better than the images. Thus, we designed the tasks such that success could be mostly determined by a static image instead of a sequence of frames.

To help GPT4-V reason over the images, we experimented with some visual overlays on the input as guidelines for the model to make its judgement (Figure 3). For example, in the CartPole task, the blue dot at the center of the cart should intersect the dotted red line and the pole itself should be colinear with the line. However, these visual cues did not ultimately result in better performance.

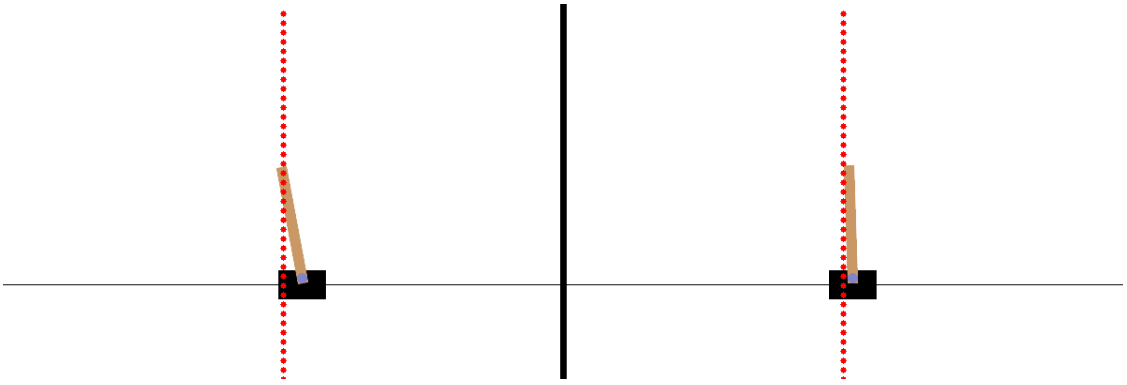

Figure 3: VLM Visual Input for CartPole
The black bar is intended to help the model differentiate between the left and right image, and the red dotted line is used as a suggestion for optimal pole alignment.

Finally, we also used chain-of-thought prompting (Wei et al., 2022) to encourage GPT4-V to output reasonable ratings. We found this works very well for the Walker2d splits task but doesn't help for CartPole.

4 RESULTS

4.1 CARTPOLE

Our objective with the CartPole environment was to replicate the original reward from preferences, i.e. we wanted to use feedback to teach the cart how to balance the pole. In Figure 4, we show that rewards learned from 350 human preferences are sufficient for an RL algorithm like PPO to solve CartPole. Unfortunately, when testing with GPT4-V, we were unable to get any sort of usable ratings from the model as the LLM heavily biased toward outputting LEFT.

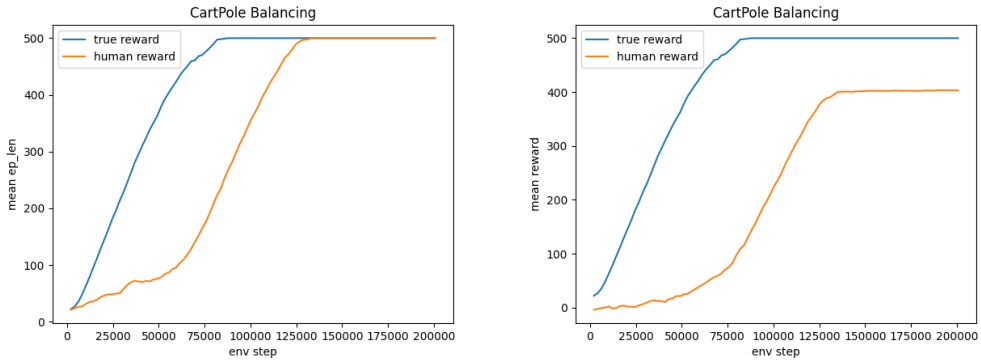

Figure 4: CartPole episode length vs step (left) and reward value vs step (right). Blue curves use baseline reward function from Gym. Orange curves use learned reward function from human preferences.

From the left plot, we observe that our model performs as well as the baseline model in the long run but learns slower. This is expected as the baseline starts with a validated reward function while our model needs time to learn a good reward function.

The baseline Gym reward function is simply the episode length as the goal is to keep the pole balanced for as long as possible and each trajectory terminates as soon as the pole falls. From the right plot, we see that our model learns essentially the same reward function after 100k steps, only shifted down by 100.

4.2 LUNARLANDER

Our objective with the LunarLander environment is to place the lander in the upper left corner of the frame and have it hover there. Unlike CartPole, we didn't have a quantifiable measure of the performance of the model so instead we link a video of the policy trained from 350 human labels. We also include the training/validation loss and accuracy curves of the reward model as it learns from preference data in Figure 5. Similar to CartPole, we could not get any usable ratings from GPT4-V as it mainly outputted LEFT.

4.3 WALKER2D

Our objective with the Walker2d environment was to do the splits. Surprisingly, GPT4 was more able to output reasonable answers when asked for ratings compared to the CartPole and LunarLander environments. One potential explanation is that the MuJoCo environments have less objects and are arguably simpler to reason about, even though they're harder to control. It's not obvious that the purple shape in LunarLander is a spaceship and the orientation is also hard to judge.

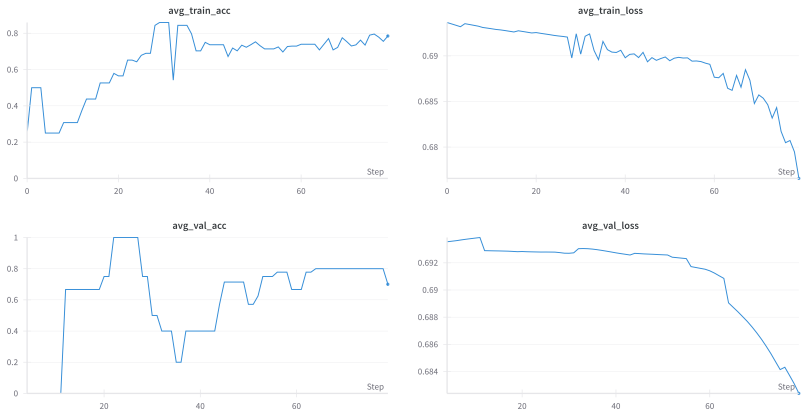

Figure 5: Reward model training curves from the LunarLander environment
In general, we found that validation accuracy $> 75\%$ is necessary for good performance

With Walker2d, we use GPT4-V and chain-of-thought prompting to elicit 380 AI ratings. Performance is shown in Figure 1 and the training plots are shown in Figure 6. Also surprisingly, even though validation performance is abysmal as shown in the graphs, the reward predictor is good enough to be used by an agent to perform the desired task.

Figure 6: Reward model training curves from the Walker2d environment.

5 Limitations and Future Work

Although preference training can be used help align an agent's current behavior to a task, we found it was challenging to use it to direct an agent towards novel behavior. For example, training an agent to write English letters while only being able to provide preference feedback would be quite challenging as the model would need randomly explore and connect a lot of sequences together for good segments to be sent to the preference labeler.

In the future, we hope to explore leveraging short video segments rather than static images when labeling preferences. This would help the reward model capture the reliance that complex tasks often have on a sequence of actions which is lost when resorting to a single timestep.

6 CONCLUSION

Overall, we found that although current VLMs aren't able to robustly perform the kind of visual reasoning necessary for preference rating, they can still be used in certain settings to reduce human labor and train nontrivial tasks for an agent. As the capabilities of VLMs continue to improve, their ability to distill and transfer their world knowledge into agents or smaller models will also improve, allowing for complex tasks requiring general knowledge to be more accessible for researchers and engineers without large training budgets. Much of our training was run on single GPUs or CPUs, with less than $10 of OpenAI credits. We look forward to the innovation that is soon to be built on top of these foundational models.

AUTHOR CONTRIBUTIONS

Bryce: Worked on initial reward model design and policy training loop. Ran human-labeled (baseline) experiments. Drafted introduction and design sections of writeup.

Jonathan: Worked on the multiprocessing loop and policy/reward training scripts. Ran and proposed experiments and collected data for human preference runs. GPT4-Vision prompt tuning and optimization. Added experimental section of writeups.

Laryn: Human & LLM preference interface design and implementation, running human-labeled (baseline) experiments, debugging multiprocessing & reward training code, generating write-up plots.

7 APPENDIX

7.1 CODE

GitHub

7.2 LLM PROMPTS

Environment	Task	Prompt
Swimmer	Make the letter M	You are an expert in the English language. This is an image of two lines on a checkboard background. Which of the two lines looks more like an M. They both probably won't look much like the letter but give your best judgement. Output a single answer: LEFT or RIGHT.
CartPole	Balance the pole	You have been given two drawings of a square cart trying to balance a brown, wooden pole. Priority 1: The pole should be as close to vertical as possible, perpendicular to the ground. Priority 2: the cart should be centered in its respective frame. Pick the image (LEFT or RIGHT) that best satisfies these priorities. They both might bad at balancing the pole but give your best judgement. Output a single answer: LEFT or RIGHT.
LunarLander	Hover in the top left corner	You have been given two drawings separated by a red line. They each contain a purple spaceship in the air. The spaceship should move to and hover at the upper left corner. Pick the image (RIGHT or LEFT) where the purple shape is closest to the upper left corner of its frame. Do not pick the image if the spaceship is not visible in frame. Do not pick the image if the purple trapezoid spaceship is pointed towards the ground. They both might be far away, but give your best judgement about which is most likely to arrive at the corner. Output a single answer: RIGHT or LEFT.
Walker2D	Do the splits	You have been given two images of an object with two legs in a checkerboard background. Pick the image (RIGHT or LEFT) where the object's legs look most like they're doing a split, specifically, the purple and brown legs should be far apart. They both might not look like a split, but give your best judgement about which is most similar to a split. Explain your reasoning step-by-step and end with a single answer: RIGHT or LEFT.

REFERENCES

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning, 2022.

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

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

Yuqing Du, Ksenia Konyushkova, Misha Denil, Akhil Raju, Jessica Landon, Felix Hill, Nando de Freitas, and Serkan Cabi. Vision-language models as success detectors, 2023.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2022.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network, 2015.