RL-VLM-F: Reinforcement Learning from Vision Language Foundation Model Feedback

Colin Czarnik[1], **Carter Korzenowski**[1], **Layne Malek**[1], **TJ Neuenfeldt**[1], **Jehan Patel**[1], **Arthur Yang**[1]

[1]University of Michigan
{cczar, cakorzen, lmalek, tjneu, pajehan, yarthur}@umich.edu
Group 11

Abstract

Learning reward functions within robotics has long been a challenge in the Reinforcement Learning (RL) community. Recent work by Wang et al. (RL-VLM-F) demonstrates that vision-language foundation models (VLMs), conditioned on task descriptions and visual observations, can provide informative feedback to guide policy learning. However, in most robotic applications, the policy being learned is under multiple constraints. RL-VLM-F suffers as a result of this: the image can only provide so much information to the VLM.

In this paper, we propose an extension to RL-VLM-F by providing a robot's metadata to the VLM to improve the reward function being learned. In addition, we explore tournament-style preference labeling to support faster convergence of the policy being learned. We demonstrate the successful replication of RL-VLM-F as well as the results of these two extensions. We find that the tournament-style preference labeling is faster than the original method and achieves similar success rates. Additionally, we find that incorporating metadata into the VLM feedback enables the policy to trade off success and total movement for some environments, eventually reaching success rates similar to those in the original paper.

Introduction

A longstanding challenge in the field of Reinforcement Learning (RL) is the design of reward functions that guide agents to solve tasks efficiently. Traditionally, reward engineering relies on extensive human effort, requiring manual specification of rewards or iterative trial-and-error methods. This process requires significant time and effort from human experts, significantly limiting the scalability of RL for more complex tasks and environments. Automating reward function generation would be highly beneficial; it can accelerate RL development, generalize RL to broader tasks, and reduce human error by removing reliance on hand-crafted signals.

The recent work RL-VLM-F: Reinforcement Learning from Vision Language Foundation Model Feedback addresses this issue by leveraging Vision Language Models (VLMs) to automatically update reward functions (Wang et al. 2024). Instead of manually writing and updating reward functions, RL-VLM-F queries VLMs with task descriptions and agent observations, using the model's preferences over image pairs to generate reward signals. This approach can significantly reduce the cost of reward engineering, especially for tasks with complex visual states or tasks that require semantic reasoning.

The authors of RL-VLM-F claim that their method can generate robust, preference-based reward functions solely from text descriptions of a task goal and image observations of the agent, without the need for human-supervised or ground-truth reward engineering. They find that RL-VLM-F outperforms several prior methods, including CLIP, BLIP, and RoboCLIP, in terms of policy success rates and final task performance. They perform ablations to reveal that RL-VLM-F's learned rewards are closely aligned with ground-truth task progress, and VLM-generated labels are sufficiently accurate.

Building on RL-VLM-F, we explored two extensions designed to improve convergence speed and model capabilities in RL scenarios. We first implemented tournament-style image comparisons, where we modified the VLM query to compare three images tournament-style instead of two images pairwise. We then explored completing multiple tasks simultaneously with Multi-Objective Reinforcement Learning (MORL), where we added environmental metadata into the VLM prompt in order to accomplish multiple objectives.

Link to code repository: https://github.com/eecs-498-group-11/RL-VLM-F

Related Work

Real-World Offline Reinforcement Learning from Vision Language Model Feedback. Offline reinforcement learning requires model to use a pre-collected, offline dataset to learn policies. The reward labeling of this offline dataset is often time consuming and challenging for more complex tasks. This paper builds on the work done in RL-VLM-F to create a new system that uses preference feedback from a VLM and a text description to automatically generate reward labels for offline datasets (Venkataraman et al. 2025). The method presented performs well in a real-world robot-assisted dressing task as well as in simulation tasks that manipulate rigid and deformable objects.

Davidson-Luce Model for Multi-Item Choice. Similar to our work, the Davidson-Luce (DL) Model generalizes pairwise choice models to consider more than 2 items in a single comparison, with the aim of extracting more information per query. DL models perform joint, multi-item

comparisons in a single decision step (Firth, Kosmidis, and Turner 2019). In contrast, Extension A uses a tournament-style approach to select the best item, comparing only two states at a time. This comparison structure is therefore fundamentally pairwise even when ranking multiple states.

Modeling Human Preferences Using Transformers for Reinforcement Learning. Like RL-VLM-F, this work aim to perform reinforcement learning by learning a reward function from preference signals rather than just on environment rewards. The Preference Transformer uses a transformer architecture to model human preferences over full trajectories, allowing it to capture long-range temporal dependencies and non-Markovian rewards (Kim et al. 2023). On the other hand, RL-VLM-F focuses on using language and vision-based feedback to guide reward learning and is guided by a vision language model.

Trajectory Improvement and Reward Learning from Comparative Language Feedback. This paper uses human language feedback rather than a preference based selection to learn human preferences (Yang et al. 2024). This is similar to our Extension A since we also changed the preference selection system to better learn human preferences. This paper also seeks to incorporate additional trajectory data in order to improve the trajectory of robots and learn reward functions that better learn human preferences. This is similar to our extension B where we incorporate positional metadata to help learn a reward function with more constraints.

Background

The model used in our extensions is similar to the original paper, with the only modification being the number of states compared. The standard setup for reinforcement learning with Markov Decision Processes (Sutton and Barto 2018) entails an agent interacting with its environment and receiving rewards for it, influencing future actions. In each timestep t, the environment will be in state s_t. Based on this, the agent will take action a_t, as this happens, the state will be updated to s_{t+1} and the agent will receive a reward r_t. The agent will choose actions to maximize the discounted sum of rewards, prioritizing more recent actions with a discount factor of γ in the total return: $R = \Sigma_{k=0}^{\infty} \gamma^k r(s_k, a_k)$

Preference-based Reinforcement Learning. In the original RL-VLM-F paper, a preference-based RL approach was used and built upon. In this model, an agent learns a reward function based on preference labels from prior states. (Christiano et al. 2017; Chu et al. 2023) A set of sequences is selected $(\sigma^0, ..., \sigma^{n-1})$ where a sequence is defined here as a single state (represented by a single image) $\sigma = s$ and n is the number of states being compared. A label is given for which sequence is preferred: $y \in \{-1, 0, ..., n-1\}$, where -1 corresponds to no preference, and any non-negative number corresponds to the preferred state. In this paper we consider $n = 2$ (pairwise comparison: used in original paper, replication and Extension B) and $n = 3$ (multi-state comparison: used in Extension A). We use a modified version of the Bradley-Terry Model (Bradley and Terry 1952) to calculate probabilities of preferences using a parametrized reward function r_ψ over the state labels:

$$P_\psi[\sigma^0 \succ \sigma^1, ..., \sigma^{n-1}] = \frac{\exp\left(r_\psi(s_t^0)\right)}{\sum_{i=0}^{n-1} \exp\left(r_\psi(s_t^i)\right)} \quad (1)$$

Where $\sigma^i \succ \sigma^{j \neq i}$ indicates that segment i is preferred over any other segment j. Based on a set of preference $D = \{(\sigma_i^0, ..., \sigma_i^{n-1}, y_i)\}$, the reward function r_ψ is optimized by minimizing the loss function listed below:

$$\mathcal{L}_{\text{Reward}} = -\mathbb{E}_{(\sigma^0,\ldots,\sigma^{n-1},y)\sim\mathcal{D}} \left[\sum_{i=0}^{n-1} \Big(\mathbb{I}\{y = (\sigma^i \succ \sigma^{j\neq i})\} \cdot \log P_\psi[\sigma^i \succ \sigma^{j\neq i}] \Big) \right] \quad (2)$$

PEBBLE (Lee, Smith, and Abbeel 2021) is used to train this model, using unsupervised pre-training and off-policy learning to update the reward function and policy.

Replication

Methods The goal for our replication of the RL-VLM-F paper was to replicate Figure 2, which contains the learning curves that compare the success rate of RL-VLM-F with other baseline VLM models, including the VLM score, CLIP, BLIP-2, and RoboCLIP for seven tasks. Across those tasks, RL-VLM-F outperforms every baseline model in every task and even performs up to that of the ground-truth baseline model - the reward model provided by the authors of each task - in six of the seven tasks. This graph stands out because it shows how much more capable RL-VLM-F is at generating an accurate reward function compared to some of the other most effective baseline algorithms, and thus we decided to make it the focus of our replication.

The authors of this paper provided a code repository containing all the code they used to train the model, all of the resources for creating and displaying the environment, as well as a set of cached labels that can be used to train the generation of the reward model without actually spending resources to query an API to generate new labels. For the purposes of the replication and the additional cost constraints for querying an API, the cached labels are sufficient for this replication. We decided to only replicate the parts of the figure that are the most productive given our limited time and resource constraints. The seven tasks can be categorized as using the following simulated physics environments:

- MetaWorld (Rigid object articulation and manipulation) (Yu et al. 2020): Sweep Into, Soccer, Drawer Open
- SoftGym (Deformable object manipulation) (Lin et al. 2021): Cloth Fold, Pass Water, Straighten Rope
- OpenAI Gym: Cartpole

Each of these environments required their own setup to get running: SoftGym required using old Docker images and had an extensive setup, whereas MetaWorld and OpenAI

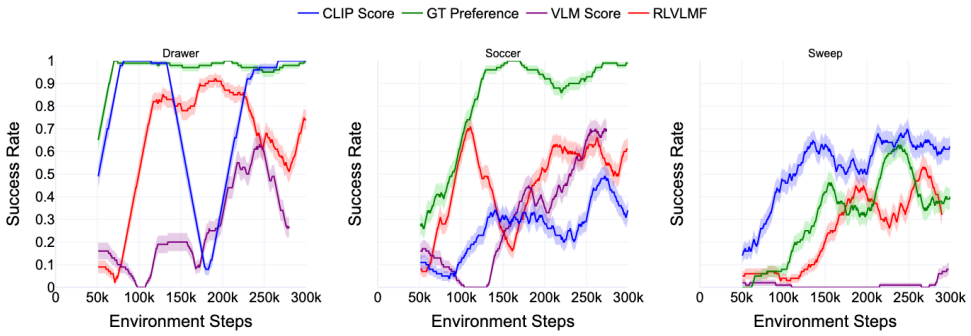

Figure 1: Replicated learning curves of all MetaWorld environments trained to 300k iterations using a single seed and rolling window of 100 iterations.

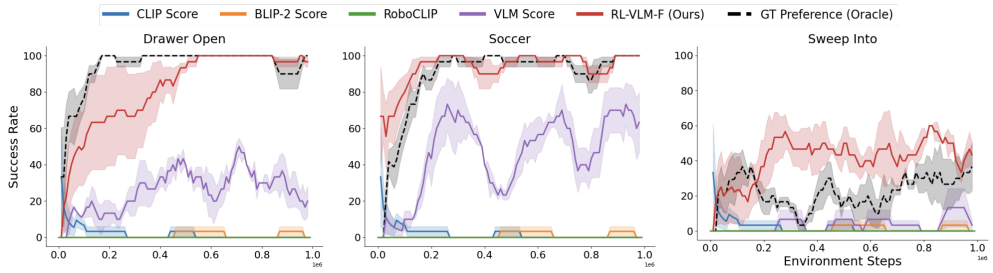

Figure 2: Learning curves of all MetaWorld environments in the original paper. BLIP-2 Score and RoboCLIP were not used in our replications since they showed poor results in the original paper.

Gym simply involved setting up a virtual environment. We concluded that it would be in our best interests to replicate just one of the three classes of tasks. We decided to focus on the three MetaWorld tasks as they were the easiest to set up and offered the most flexibility when it comes to changing the training algorithm for extensions.

In Figure 2, each model was trained for one million environment steps in all seven tasks with an average success rate over several seeds. However, because the maximum time we could train any given model for was bounded to eight hours, training for one million steps was infeasible, as the training model reached only 250k-300k iterations in those eight hours. However, the model for each MetaWorld task began to converge after just 300,000 environment steps in the majority of cases, which was feasible within the eight-hour time limit. Therefore, training each model up to just that point in the graphs was still sufficient to show most trends up to the point where they converged. In addition, we only trained each environment on one seed and only used the following metrics for our comparison due to time constraints: RL-VLM-F, VLM Score, CLIP, and ground truth. We chose these metrics from the original figure as ground truth provided a good upper-bound, the VLM score provided a good lower-bound, and CLIP is another useful reference that we expect RL-VLM-F to consistently perform well against.

Results and Discussion As shown in Figure 1, the first 300k iterations closely resemble that of the original figure from the paper in Figure 2. One observable difference in the Drawer Open example is that the CLIP score dips severely during the middle of the run. Similarly, in the Soccer environment, the RL-VLM-F metric dips. Our hypothesis is that this is caused because we used a single seed and only used cached labels for our replication. There was less data in the cached labels and only one seed was run, which could have lead to this drop in success. However, we weren't able to evaluate this hypothesis due to time and resource constraints, so more testing will be needed to identify the exact cause.

The Drawer Open example achieved a similar success rate for RL-VLM-F and VLM score compared to the paper. The success rate for RL-VLM-F achieved a lower rate than the original. The Sweep Into example achieved a similar success rate for RL-VLM-F and had a high CLIP score. The variation between these plots and the ones originally presented in the paper are most likely due to using a single seed. Since the authors averaged five seeds, spikes will be smoothed out compared to what we see in the Drawer and Soccer examples.

Extensions

Two extensions were provided: comparing three images using tournament-style comparison and providing robot metadata to the VLM. Compared to the replication part where cached labels were used, the Gemini API was used. The authors originally chose Gemini 1.0 models; these extensions used `Gemini-2.5-flash-lite` for the VLM and

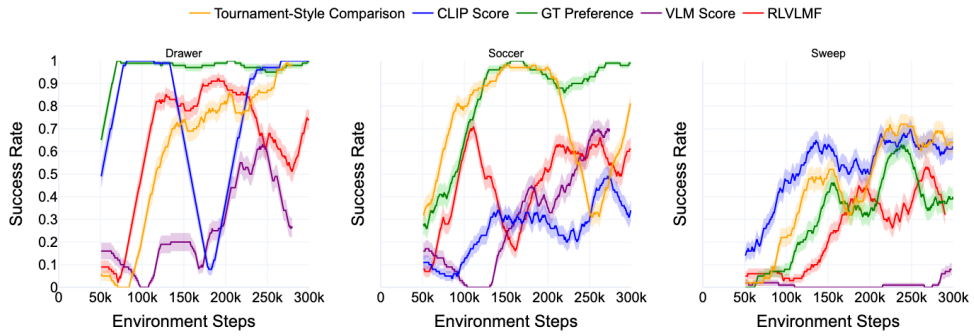

Figure 3: Learning curves of all MetaWorld environments trained to 300k iterations using a single seed and rolling window of 100 iterations using Tournament-Style Comparison.

`Gemini-2.5-pro` for the second-stage LLM as they are the latest models and may perform better than 1.0 models.

Extension A: Tournament-Style Comparison

Our intuition with Extension A is based on the observation that RL-VLM-F's original pairwise comparison strategy is inherently limited: by only comparing two states at a time and selecting a single preferred direction, the model requires many iterations to discover optimal state patterns.

We hypothesize that by expanding the comparison set (prompting the VLM to rank more than two states simultaneously), the model will receive a more informative signal about the overall structure of the task space. This tournament-style ranking (performing two back-to-back comparisons, with the preferred state in the first comparison being compared against the third state) allows the RL agent to learn from more informative feedback, potentially accelerating convergence towards higher-quality reward functions.

Methods This extension required several modifications to the original paper's framework. Because this involves moving from a two-state to a three-state comparison, we first updated the reward model and the Bradley-Terry probability computation (see Equations 1 & 2) to include a third state and produce a final preference label over all three images.

First, we had to modify both the reward model and the reward model score. In these files, we had to update all parts of the code with pairwise comparisons to handle the comparison of three images to each other. We did this by adding a third sequence of state-action pairs, a third reward segment, and a third image slice, duplicating the original logic to create these components and applying it to a third option for each component. Additionally, we added a third segment to the replay buffer. Finally, we added a third preference label instead of using the former binary system of preference labeling.

In order to get a preference that accounts for the third image, we modified the prompt file inputted into Gemini. We added prompts such as "What is shown in Image 3?" and added an additional comparison step to compare the winner from step 1 and image 3, and after implementing this comparison, we prompted Gemini to output the winner of the three (returns 0, 1, or 2 for images 1, 2, or 3 respectively).

Results and Discussion The plots comparing the success rates of Tournament-Style Comparison to Pairwise Comparison can be seen in Figure 3. The Soccer environment, based on the seed, has a great learning curve. It achieved a success rate similar to the ground truth and exceeds other baselines (including the original paper's pairwise comparison). The model could only be trained until iteration 250k. After this, it was rerun using the saved model from the most recent iteration. There's an interesting dip at 250k in the graph, even though it was using a saved model. The most likely cause for this is the fact that the training model uses a replay buffer to sample its images. When the model is rerun, this buffer is wiped out, causing poor performance for the first couple thousand iterations. The Sweep environment performed better for the most part compared to the original paper's algorithm. The Drawer environment tournament-style comparison performed quite similar to the original paper and even outperformed it towards the end. Future steps could include more rigorous testing.

Given more time, more seeds should be tested for longer to more accurately replicate the RL-VLM-F paper. Given more images to compare, it seems to learn the environment quicker and have a higher success rate sooner. Of course, the more images provided comes with diminishing returns. Similar to humans, when presented with many images and having to pick a preference, the job becomes much more difficult. Three or four images is manageable given this tournament-style comparison, but anymore images may lead to worse results. Therefore, further testing needs to be conducted to answer this.

Extension B: MORL with Metadata

A limitation of the model posed in the original paper is that the tasks are rated pure on completion, with no basis for how the task was completed. The goal of our second extension was to modify the tasks to increase complexity to see if RL-VLM-F can handle bigger problems. There were multiple potential approaches to this, such as tasks that have multiple stages and sampling more than one image to gather a move-

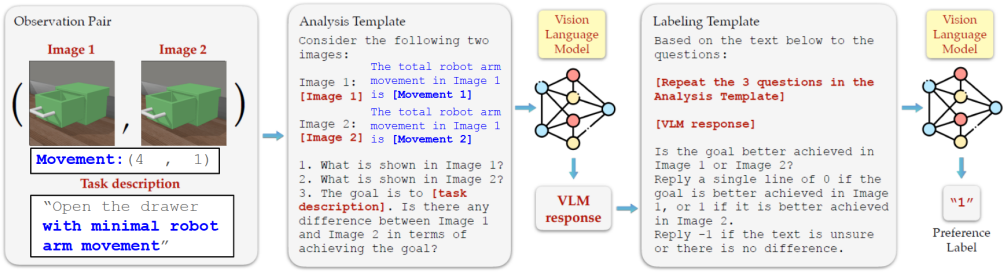

Figure 4: Modified prompting structure to include metadata, with changes in blue text. Task description includes secondary goal of reducing arm movement. Based on the prompt structure from the original RL-VLM-F paper.

ment in a state. What we settled on was a focus on efficiency using MORL.

Multi-Objective Reinforcement Learning, as the name implies, involves optimizing multiple rewards simultaneously. Our goal of using MORL to extend RL-VLM-F is to complete the tasks in a more efficient way by adding an additional objective. Specifically, we wanted completion of the task as a primary objective and minimal motion in the robot arm as a secondary objective. The way we achieve this is by measuring the total movement of the robot arm in the task environment, then including that total movement number with the image when prompting the VLM to pick a preference. The task description is modified by adding that movement should be minimized. The hypothesis posed in this extension is that if we give metadata to the VLM with a modified task description, a similar success rate can be achieved with a lower amount of robot arm movement.

Methods The implementation of this approach involved a few modifications to the code of the original paper. The main change was extracting the desired metadata from the environment simulation and inputting it directly into the VLMs' prompts.

For each environment, we calculate the total arm movement over a trajectory by accumulating the displacement of the robot's arm at every timestep. This is computed directly from the simulation environment of physics via the position of the tracked robot arm, aligning with how the environment resets and updates its position. The total movement value is recorded alongside each segment of the sampled trajectory during the experience collection.

This metadata is then integrated into the preference-learning pipeline, demonstrated in Figure 4. When two trajectory segments are passed to the VLM for scoring, the prompt is modified from the original instruction (e.g. "Move the soccer ball into the goal.") to include an additional requirement emphasizing efficiency (e.g. "Move the soccer ball into the goal with minimal robot arm movement."). Along with the images, the VLM receives the total-movement values and is asked to choose with segment better satisfies the task objective under this expanded definition.

Results and Discussion The plots of the results are shown in Figure 5. In terms of the success rates of the RL-VLM-F model when using metadata compared to not using metadata, the success rate at a similar level or better for two of the environments (Drawer and Soccer) and performed worse for the other (Sweep). The VLM score generally performed worse than the replication, likely due to the difficulty the VLM has giving a numerical score for the multi-objective task. Another pitfall that was also brought up in the extension A results is that the RL-VLM-F and VLM models could only be trained between 150K and 215K iterations in one job under the restrictions we had. This resulted in us needing to rerun them, resuming from where they left off, but this caused the model to not retain the replay buffer, resulting in a significant dip in the success rate in the early iterations of the rerun.

For movement, the RL-VLM-F model using metadata had similar metrics for the Drawer and Soccer environments, while having lower movement levels for the Sweep environment. Note that lower movement is the objective of the model, so a tradeoff can be noticed here. The Sweep environment seems to have been able to take both goals into account, reducing movement, while also reducing success rate. However, near the end of the simulation after around 300K steps, the model using metadata records the same success rate with lower movement. The VLM Score metrics for movement do not show significant difference between using and not using metadata.

Conclusions

We have successfully replicated the key results of RL-VLM-F, showing that preference-based feedback from VLMs is a strong substitute for manual reward engineering. Our replication of the tasks using the MetaWorld environment shows that RL-VLM-F can outperform popular baselines such as CLIP Score and VLM Score, and is capable of achieving the performance of ground-truth rewards.

To build on this method, we proposed and evaluated two extensions to the RL-VLM-F structure. We first implemented our tournament-style multi-image ranking extension, showing that broader comparisons can achieve faster reward function convergence compared to a simpler pairwise approach. We believe that continuing to increase the number of images in each comparison will yield diminishing returns, but further research will be needed to balance complexity and information gain.

Secondly, we explored MORL by integrating environmental metadata (such as robot arm position) into the VLM prompt for learning reward functions. Our results show that

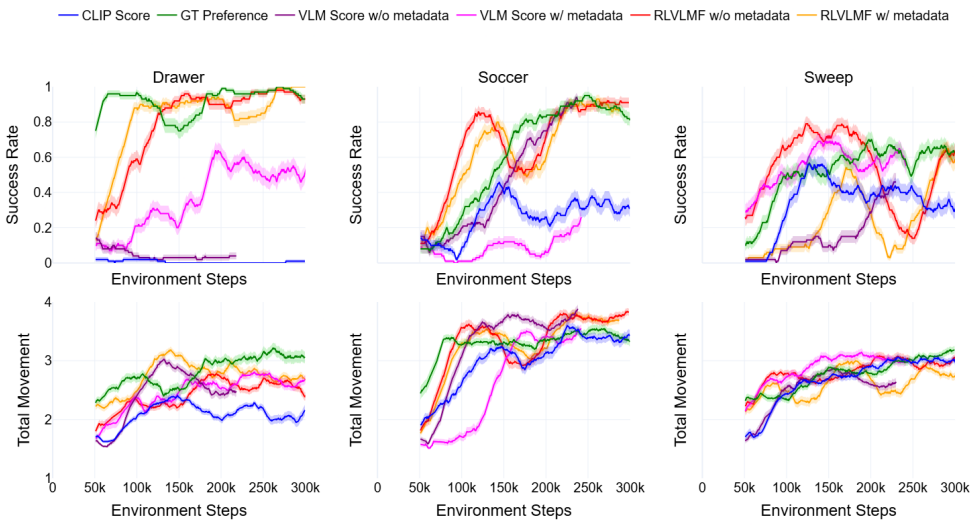

Figure 5: Success rate and total movement curves for Multi-Objective Learning.

VLMs are capable of producing preferences not only for task completion objectives but also for efficiency constraint objectives. However, our performance varied with the task environment compared to the original RL-VLM-F implementation, and we were unable to explore more complex use cases due to time and resource constraints. This approach will need further testing before it can be broadly implemented.

Given additional time and resources, there are several potential areas for continued research. For Extension A, exploring the effect of comparing different numbers of images could allow us to find the optimal comparison set size for fast policy convergence. We could extend this further by using temporal data, where we rank sequences of images rather than single snapshots; this could improve the accuracy of preference labeling by providing more precise context and by increasing the quantity of available data. Finally, it would be valuable to evaluate our Extension B's metadata-augmented approach on environments with more complex dynamics, where it would have a greater impact.

Societal Impact

The automation of reward engineering in RL through VLMs has large potential for positive impact; it can accelerate developments in robotics, healthcare, and manufacturing by allowing intelligent agents to acquire complex skills with minimal human effort. By increasing accessibility to scalable RL deployment, our work can help cultivate more adaptive and accessible technologies, from household assistive robots to healthcare technologies.

As mentioned in the original paper, pre-trained VLMs are used for preference labeling. Any bias present in the VLMs may affect the outcome of the reward function and learned policy. Careful testing of the policy behavior should be used before deploying anywhere. Similarly, any images taken from environments can be used to train these VLMs. Be cautious if using in a proprietary setting.

References

Bradley, R. A.; and Terry, M. E. 1952. Rank Analysis of Incomplete Block Designs: The Method of Paired Comparisons. *Biometrika*, 39(3-4): 324–345.

Christiano, P. F.; Leike, J.; Brown, T.; Martic, M.; Legg, S.; and Amodei, D. 2017. Deep Reinforcement Learning from Human Preferences. In *Advances in Neural Information Processing Systems*, volume 30.

Chu, K.; Zhao, X.; Weber, C.; Li, M.; and Wermter, S. 2023. Accelerating Reinforcement Learning of Robotic Manipulations via Feedback from Large Language Models. arXiv:2311.02379.

Firth, D.; Kosmidis, I.; and Turner, H. 2019. Davidson-Luce model for multi-item choice with ties.

Kim, C.; Park, J.; Shin, J.; Lee, H.; Abbeel, P.; and Lee, K. 2023. Preference Transformer: Modeling Human Preferences using Transformers for RL. In *International Conference on Learning Representations*.

Lee, K.; Smith, L. M.; and Abbeel, P. 2021. PEBBLE: Feedback-Efficient Interactive Reinforcement Learning via Relabeling Experience and Unsupervised Pre-training. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*.

Lin, X.; Wang, Y.; Olkin, J.; and Held, D. 2021. SoftGym: Benchmarking Deep Reinforcement Learning for Deformable Object Manipulation. In *Proceedings of the 2020 Conference on Robot Learning (CoRL)*, volume 155 of *Proceedings of Machine Learning Research*.

Sutton, R. S.; and Barto, A. G. 2018. Reinforcement learning: An introduction.

Venkataraman, S.; Wang, Y.; Wang, Z.; Ravie, N. S.; Erickson, Z.; and Held, D. 2025. Real-World Offline Reinforcement Learning from Vision Language Model Feedback. arXiv:2411.05273.

Wang, Y.; Sun, Z.; Zhang, J.; Xian, Z.; Biyik, E.; Held, D.; and Erickson, Z. 2024. RL-VLM-F: Reinforcement Learning from Vision Language Foundation Model Feedback. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24.

Yang, Z.; Jun, M.; Tien, J.; Russell, S. J.; Dragan, A.; and Biyik, E. 2024. Trajectory Improvement and Reward Learning from Comparative Language Feedback. In *8th Conference on Robot Learning (CoRL 2024)*.

Yu, T.; Quillen, D.; He, Z.; Julian, R.; Hausman, K.; Finn, C.; and Levine, S. 2020. Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning. In *Proceedings of the Conference on Robot Learning*, Proceedings of Machine Learning Research.

Individual Contributions

Colin Czarnik

Modified prompt and model scripts to incorporate metadata for Extension B. Assisted in adapting code to function in Great Lakes for replication and extension. Ran environments for Extension B and replication. Created plot scripts for displaying results.

Carter Korzenowski

Researched into several possible tasks and approaches in order to best implement the multi-objective approach in Extension B. Modified reward model and reward model score for Extension B.

Layne Malek

Modified the code for reward model score for extension A. Modified the code for reward model and reward model score for extension B. Trained many metrics for extension A and B. Modified and updated plotting code.

TJ Neuenfeldt

Modified the code for Extension A. In addition, trained the environments/metrics for Extension A as well as some metrics for Extension B. Ran the replication environments and generated graphs for this paper.

Jehan Patel

Set up Great Lakes environment for original paper replication. Fine-tuned environment configurations for replication and Extension A. Tested model results with varying evaluation metrics from related works, original paper, and Extension A. Researched evaluation metrics for Extension B.

Arthur Yang

Implemented the calculation of the total end-effector movement as metadata for Extension B, including integrating the measure into the reward comparison pipeline. Authored documentation for Extension A explaining the intuition and expected behavior behind the methods.