Abstract

Reward engineering has long been a challenge in Reinforcement Learning (RL) research, as it often requires extensive human effort and iterative processes of trial-and-error to design effective reward functions. In this paper, we propose RL-VLM-F, a method that automatically generates reward functions for agents to learn new tasks, using only a text description of the task goal and the agent’s visual observations, by leveraging feedbacks from vision language foundation models (VLMs). The key to our approach is to query these models to give preferences over pairs of the agent’s image observations based on the text description of the task goal, and then learn a reward function from the preference labels, rather than directly prompting these models to output a raw reward score, which can be noisy and inconsistent. We demonstrate that RL-VLM-F successfully produces effective rewards and policies across various domains — including classic control, as well as manipulation of rigid, articulated, and deformable objects — without the need for human supervision, outperforming prior methods that use large pretrained models for reward generation under the same assumptions.

1. Introduction

One of the key challenges of applying reinforcement learning (RL) is designing an appropriate reward function that will lead to the desired behavior. This procedure, known as reward engineering, demands considerable human effort and trial-and-error iterations, but is often required for good results (Laud, 2004; Silver et al., 2016; OpenAI et al., 2019; Gupta et al., 2022). In this work, we aim to develop a fully automated system that can generate a reward function and use it to teach agents to perform a task with RL by using only a language description of the task, eliminating the extensive human effort required to craft reward functions manually.

Prior work has studied replacing human supervision by prompting large language models (LLMs) to write code-based reward functions (Xie et al., 2023; Ma et al., 2023b; Wang et al., 2023). However, these methods usually assume access to the environment code, rely on the low-level ground-truth state information for reward generation, and face challenges with scaling up to high-dimensional environments and observations, such as manipulating complex deformable objects.

Another related line of work obtains rewards from visual observations by using contrastively trained vision language models, such as CLIP (Radford et al., 2021), to align image or video observations with task descriptions in a learned latent space (Cui et al., 2022b; Mahmoudieh et al., 2022; Ma et al., 2023a; Sontakke et al., 2023; Adeniji et al., 2023; Rocamonde et al., 2023). However, the reward signals produced in these works are often high variance and noisy, as these CLIP-style models are limited to training exclusively on datasets where the language description directly captions the visual input; these types of datasets are not sufficiently large or diverse to allow the models to reason about tasks that are visually dissimilar to the datasets they were trained on. As a result, prior work often has to fine-tune these CLIP-style models for their specific tasks at hand (Ma et al., 2023a; Mahmoudieh et al., 2022).

To this end, we present RL-VLM-F, a method that automatically generates reward functions for agents to learn new task. RL-VLM-F (Figure 1) requires only a single text description of the task goal and the agent’s visual observations, leveraging vision language foundation models (VLMs) that are trained on diverse, general text and image corpora (e.g., GPT-4V (OpenAI, 2023), Gemini (Team et al., 2023)) instead of the limited vision-language aligned datasets used in CLIP-style models. The key to our approach is to query these models to give preferences over pairs of the agent’s image observations based on the text description of the task goal and then learn a reward function from the preference labels, rather than directly prompting these models to output a
Figure 1. RL-VLM-F automatically generates reward functions for policy learning on new tasks, using only a text description of the task goal and the agent’s visual observations. The key to RL-VLM-F is to query VLMs to give preferences over pairs of the agent’s image observations based on the text description of the task goal, and then learn a reward function from the preference labels.

raw reward score, which can be noisy and inconsistent (Sontakke et al., 2023; Rocamonde et al., 2023). This allows us to draw from the rich literature on reinforcement learning from human preferences (Christiano et al., 2017; Wirth et al., 2017; Lee et al., 2021a), without requiring actual humans, to train reward functions automatically for new tasks. We test our method on 7 tasks involving classic control, rigid, articulated, and deformable object manipulation. We show that our approach can produce reward functions that lead to policies that solve diverse tasks, and our approach substantially outperforms prior methods and alternative ways to use VLMs to generate rewards. We also perform extensive analysis and ablation studies to provide insights into RL-VLM-F’s learning procedure and performance gains.

In summary, we make the following contributions:

- We propose RL-VLM-F, a method that automatically generates reward functions for agents to learn new tasks, using only a text description of the task goal and the agent’s visual observations, eliminating the extensive human effort involved in manually crafting reward functions.
- We show that RL-VLM-F can be used to generate reward functions and learn policies that can solve a series of rigid, articulated, and deformable object manipulation tasks, and it greatly outperforms prior methods.
- We perform extensive analysis and ablation studies to provide insights into RL-VLM-F’s learning procedure and performance gains.

2. Related Works

Inverse Reinforcement Learning. Similar to our work, inverse reinforcement learning (IRL) aims at learning a reward function that can be used to learn policies for solving a task. IRL methods usually learn a reward function from expert demonstrations (Ng & Russell, 2000; Abbeel & Ng, 2004; Ziebart et al., 2008; Ho & Ermon, 2016; Fu et al., 2018; Ni et al., 2021). In contrast, while RL-VLM-F also learns a reward function to train a policy, it only requires a text description of the task goal and does not require collecting expert demonstrations.

Learning from Human Feedback. Another line of work directly learns a reward function from human feedback, in the form of pairwise trajectory preference or ranking comparisons, to train a reward function (Christiano et al., 2017; Wirth et al., 2017; Ibarz et al., 2018; Leike et al., 2018; Biyik et al., 2019; 2020; Lee et al., 2021a; Myers et al., 2021; Bıyık et al., 2022). In most cases, human preferences and rankings of robot trajectories are easier to collect than demonstrations of robot trajectories. However, because each comparison conveys little information on its own, many preference queries are needed before the reward function is well-trained enough to train an agent to perform the task. RL-VLM-F instead queries a VLM to perform the comparison to train a reward function, removing the need for extensive human labor in giving preference labels.

Large Pre-trained Models as Reward Functions. Kwon et al. (2023) first demonstrated that large pre-trained models—large language models (LLM) specifically—can generate rewards for RL agents in text-based tasks. Other works followed by demonstrating that LLMs can write structured code for training robots (Yu et al., 2023) or directly write Python code for training many kinds of agents (Xie et al., 2023; Ma et al., 2023b; Wang et al., 2023). However, many tasks are challenging to write reward functions for.
For example, cloth folding requires tracking the locations of many individual cloth keypoints, which can change from one folding task to another. In these instances, visual reasoning is better suited for understanding how to reward the agent. RL-VLM-F queries a VLM to compare agent observation images so that it can use visual observations to reason about how well the agent is progressing in a task. In addition, prior methods usually assume access to the environment code when writing the reward functions, whereas our method does not require such assumptions.

Another line of prior works rewards agents from image observations by aligning agent trajectory images with task language descriptions or demonstrations with contrastively trained visual language models (Cui et al., 2022a; Fan et al., 2022; Nottingham et al., 2023; Ma et al., 2023a; Sontakke et al., 2023; Rocamonde et al., 2023; Nam et al., 2023). However, experiments from these papers directly demonstrate that contrastive alignment is noisy and its accuracy relies heavily on the input task specification and how well-aligned the agent observations are to the pre-training data (Ma et al., 2023a; Sontakke et al., 2023; Rocamonde et al., 2023; Nam et al., 2023). These contrastively trained models also cannot explicitly reason about the task; our approach with vision-enabled large language models is more effective as the VLMs are trained on more extensive and diverse vision datasets, allowing them to interpret and reason about agent observation images for a broader range of tasks. Further, CLIP-style models have thus far been limited to outputting noisy raw scores. We demonstrate that using preferences results in superior performance to outputting raw scores, shown in our experiments in Section 6. Finally, our work shares a similar idea to RLAIIF (Bai et al., 2022), which proposed to mix preference labels generated by LLM and human in the context of fine-tuning LLMs. In contrast, we use a VLM to generate the preference labels without any human labeling, and focus on the domain of robotics control and manipulation.

3. Background

We consider the standard Markov Decision Process and reinforcement learning setup (Sutton & Barto, 2018). At every timestep $t$, the agent receives a state $s_t$ from the environment and chooses an action $a_t$ based on a policy $\pi(a_t | s_t)$. The environment gives a reward $r_t$ after the agents executes action $a_t$ and transitions to $s_{t+1}$. The goal of the agent is to maximize the return, which is defined as discounted sum of rewards $R = \sum_{k=0}^{\infty} \gamma^k r(s_k, a_k)$ with discount factor $\gamma$.

Preference-based reinforcement learning: Our work builds upon preference-based RL, in which an agent learns a reward function from preference labels over its behaviors (Christiano et al., 2017; Ibarz et al., 2018; Lee et al., 2021a;b). Formally, a segment $\sigma$ is a sequence of states $\{s_1, ..., s_H\}$, $H \geq 1$. In this paper we consider the case where the segment is represented using a single image, i.e., $H = 1$. Given a pair of segments $(\sigma^0, \sigma^1)$, an annotator gives a feedback label $y$ indicating which segment is preferred: $y \in \{-1, 0, 1\}$, where 0 indicates the first segment $\sigma^1$ is preferred, 1 indicates the second segment is preferred, and $-1$ indicates they are incomparable or equally preferable. Given a parameterized reward function $r_\psi$ over the states, we follow the standard Bradley-Terry model (Bradley & Terry, 1952) to compute the preference probability of a pair of segments:

$$P_\psi[\sigma^1 \succ \sigma^0] = \frac{\exp\left(\sum_{t=1}^{H} r_\psi(s_t^1)\right)}{\sum_{i \in \{0, 1\}} \exp\left(\sum_{t=1}^{H} r_\psi(s_t^i)\right)},$$

(1)

where $\sigma^i \succ \sigma^j$ denotes segment $i$ is preferred to segment $j$. Given a dataset of preferences $D = \{(\sigma^0_i, \sigma^1_i, y_i)\}$, preference-based RL algorithms optimize the reward function $r_\psi$ by minimizing the following loss:

$$L_{\text{Reward}} = -\mathbb{E}_{(\sigma^0, \sigma^1, y) \sim D} \left[ I\{y = (\sigma^0 \succ \sigma^1)\} \log P_\psi[\sigma^0 \succ \sigma^1] + I\{y = (\sigma^1 \succ \sigma^0)\} \log P_\psi[\sigma^1 \succ \sigma^0]\right].$$

(2)

In preference-based RL algorithms, a policy $\pi_\theta$ and reward function $r_\psi$ are updated alternatively: the reward function is updated with a dataset of preferences as described above, and the policy is updated with respect to this learned reward function using standard reinforcement learning algorithms. Specifically, we use PEBBLE (Lee et al., 2021a), a preference-based RL method with unsupervised pre-training and off-policy learning, as the underlying preference-based RL algorithm.

4. Assumptions

We make the following assumptions on the VLMs to be used in this paper: 1) We assume that the VLMs have been trained on diverse text and image corpora, enabling them to generalize well and reason across various environments and tasks. 2) The VLMs should be capable of processing multiple images simultaneously and performing comparative analyses on pairs of images as this is crucial for generating preference labels. 3) RL-VLM-F is designed to operate on static images, rather than videos. Consequently, we focus on tasks for which the quality or success of a state can be discerned from a single image or a sequence of images. We consider large pretrained vision-language foundation models, such as Gemini (Team et al., 2023) and GPT-4 Vision (OpenAI, 2023), to satisfy these assumptions.
Figure 2. We use a two-stage VLM-querying process for generating preference labels to train the reward function. In the analysis stage, we query the VLM to generate free-form responses describing and comparing how well each of the two image observations achieves the task goal. Then, in the labeling stage, we prompt the VLM with the VLM-generated text responses from the first stage to extract a preference label between the two image observations. The template shown here is the actual entire template we use for all experiments.

Algorithm 1 RL-VLM-F

```
1: // Initialize
2: Initialize policy \( \pi_\theta \) and reward \( r_\psi \)
3: Initialize the preference buffer \( D \leftarrow \emptyset \), RL replay buffer \( B \leftarrow \emptyset \), policy gradient update steps \( \mathcal{N}_r \), reward gradient update steps \( \mathcal{N}_r \), VLM query frequency \( K \), number of preference queries per time \( M \)
4: // Policy Learning and Data Collection
5: // for each iteration \( \text{iter} \) do
6: Collect state \( s_{t+1} \), image \( I_{t+1} \) by taking \( a_t \sim \pi_\theta(a_t|s_t) \)
7: Add transition \( B \leftarrow B \cup \{(s_t, a_t, s_{t+1}, r_\psi(s_t))\} \)
8: Add image observation \( I \leftarrow I \cup \{I_{t+1}\} \)
9: end for
10: // Preference by VLM and Reward Learning
11: // for each \( m \) do
12: Randomly sample two images \( (\sigma^0, \sigma^1) \) from buffer \( I \)
13: Query VLM with \( (\sigma^0, \sigma^1) \) and task goal \( l \) for label \( y \)
14: Store preference \( D \leftarrow D \cup \{(\sigma^0, \sigma^1, y)\} \)
15: end for
16: // for each \( n \) do
17: Sample minibatch \( \{(\sigma^0, \sigma^1, y)\}_{j=1}^{D} \sim D \)
18: Optimize \( r_\psi \) in Equation (2) with respect to \( \psi \)
19: Relabel entire replay buffer \( B \) using updated \( r_\psi \)
20: end if
21: end for
```

5. Method

Figure 1 provides an overview of RL-VLM-F. Unlike previous preference-based RL algorithms that require a human annotator to give the preference labels, RL-VLM-F leverages a VLM to do so based solely on a text description of the task’s goal, thus automating preference-based RL and mitigating the time-intensive human supervision required in writing reward functions or providing preference labels. RL-VLM-F works as follows: first, the policy \( \pi_\theta \) and the reward function \( r_\psi \) are randomly initialized. Given a task goal description, our method then iterates through the following cycle: (1) The policy \( \pi_\theta \) is updated using RL with the reward function \( r_\psi \), interacts with the environment, and stores image observations into a buffer; (2) A batch of image pairs is randomly sampled from the stored buffer and sent to a VLM. The VLM is queried to produce preference labels for these image pairs in terms of which one better performs the task based on the text description of the task goal; (3) The reward model is updated with the loss in Equation (2) using the preference labels produced by the VLM. The full detailed procedure of RL-VLM-F can be found in Algorithm 1.

5.1. Prompting VLMs to generate preference labels for reward learning

To train the reward model \( r_\psi \), we first need to generate preference labels from the VLM. To do this, we sample two images from the “image observation buffer” \( I \), which stores image observations of the policy during learning, and then query the VLM for which of the two images better performs the task according to the text goal description (Algorithm 1 lines 17-18).

The querying process is illustrated in Figure 2. It consists of two stages: an analysis stage and then a labeling stage. In the analysis stage, we query the VLM to generate free-form responses describing and comparing how well each of the two images achieves the task goal. Then, in the labeling stage, we prompt the VLM with the VLM-generated text responses from the first stage to extract a preference label between the two images.\(^1\) Specifically, the labeling stage

\(^1\)We can also use an LLM in this stage as it only requires text inputs, but for simplicity, we use the same model as for the first
prompt repeats the questions in the analysis prompt, fills in the VLM’s response from the analysis stage, and then asks the VLM to generate a preference label \( y \in \{-1, 0, 1\} \). We specify in the prompt that 0 or 1 indicates that the first or second image is better, respectively, and -1 indicates no discernible differences. We do not use the image pairs to train the reward model if the VLM returns -1 as the preference label. Finally, as shown at line 19 of Algorithm 1, we store the preference labels produced by the VLM into the preference label buffer \( D \) during the training process. Standard preference-based reward learning can then be performed (as detailed in Section 3) to train the reward function with Equation 2 using the preference buffer \( D \). Reward learning corresponds to lines 21-24 in Algorithm 1.

To minimize prompt engineering effort, we use a unified template across all environments (the exact entire template is shown in Figure 2). Therefore, to train a policy for a new environment with RL-VLM-F, one only needs to provide the task goal description; the labels and subsequently the reward function will then be automatically trained with the above process.

5.2. Implementation Details

For policy training, we use SAC (Haarnoja et al., 2018) as the underlying RL algorithm. As in PEBBLE (Lee et al., 2021a), we relabel all the transitions stored in the SAC replay buffer once the reward function \( r_\psi \) is updated (line 25 in Algorithm 1). We set the policy gradient update step \( N_\pi \) to be 1. The values of all other parameters in Alg. 1 can be found in Appendix B.

6. Experiments

6.1. Setup

We evaluate RL-VLM-F on a set of tasks, spanning from straightforward classic control tasks to complex manipulation tasks involving rigid, articulated, and deformable objects. The tasks are as follows.

• One task from OpenAI Gym (Brockman et al., 2016):
  - CartPole where the goal is to balance a pole on a moving cart.

• Three rigid and articulated object manipulation tasks from MetaWorld (Yu et al., 2020) with a Sawyer robot:
  - Open Drawer, where the robot needs to pull out a drawer;
  - Soccer, where the robot needs to push a soccer ball into a goal; and
  - Sweep Into, where the robot needs to sweep a green cube into a hole on the table.

• Three deformable object manipulation tasks from Soft-stage of the querying process (a VLM).

For MetaWorld tasks, we use the author-defined task success rate of the policy as the evaluation metric (Yu et al., 2020). For all other tasks, we report the episode return of the learned policy. For all methods, the policy is learned
with state observations, and we use the same policy learning
hyper-parameters for all methods, i.e., the only difference
between all compared methods is the reward function.
For methods where a reward function needs to be learned (RL-
VLM-F and VLM Score), the reward function is learned
using image observations. For RL-VLM-F and the VLM
Score baseline, we use Gemini-Pro (Team et al., 2023) as the
VLM for all tasks except Fold Cloth. We find Gemini-Pro
to perform poorly on Fold Cloth. We find Gemini-Pro
to perform poorly on Fold Cloth, so we instead use GPT-
4V (OpenAI, 2023) as the VLM for this task for these two
methods (see Appendix E for a comparison of Gemini-Pro
and GPT-4V on this specific task). We did not run GPT-4V
on all tasks due to its quota limitations. For all methods
except RoboCLIP, we remove the robot from the image for
the MetaWorld tasks, as these tasks are all object-centric
and removing the robot allows the VLM to focus on the
target object when analyzing the images. We keep the robot
within the image for RoboCLIP following the original pa-
per’s setup. We test RoboCLIP only on the MetaWorld
tasks, as this is the set of tasks where the original method is
evaluated.

6.2. Does RL-VLM-F learn effective rewards and
policies?
We first examine if RL-VLM-F leads to useful rewards and
policies that can solve the tasks. The learning curves of all
compared methods on all tasks are shown in Figure 4. As
shown, RL-VLM-F outperforms all other baselines in all
tasks. We find that prior approaches using CLIP or BLIP-2
score can only solve the easiest task – CartPole, and strug-
gle for more complex environments, such as the rigid object
manipulation tasks in MetaWorld and the deformable object
manipulation tasks in SoftGym. The text version of Robo-
CLIP performs poorly on all three MetaWorld tasks, align-
ing with the original paper’s results, as RoboCLIP works
the best with video demonstrations available. RL-VLM-F
also outperforms VLM Score in all tasks, which indicates
that prompting VLMs to output a preference label for re-
ward learning results in better task performance in contrast
to treating the VLM as a reward function that outputs raw
reward scores. We also observe that RL-VLM-F is able to
match the performance of using GT preference in all tasks
except Cloth Fold, which suggests we can use a single text
Reinforcement Learning from Vision Language Foundation Model Feedback

Figure 5. Comparison of the achieved final state of different methods on SoftGym deformable object manipulation tasks: Fold Cloth (Top), Straighten Rope (Middle), and Pass Water (Bottom). RL-VLM-F achieves better final states compared to all the baselines.

6.3. What is the accuracy of VLM preference labeling?

Given that RL-VLM-F can learn effective rewards and policies that solve the tasks, we perform further analysis on the accuracy of the preference labels generated by a VLM. To compute accuracy, the VLM outputs \{-1, 0, 1\} (no preference, first image preferred, second image preferred) which we compare to a ground truth preference label defined according to the environment’s reward function. Note that we discard the image pairs with a label -1 (no preference) when training the reward model.

Our intuition is that, like humans, it would be hard for the VLM to give correct preference labels when comparing two similar images, and easier to produce correct preference labels when the two images are noticeably dissimilar in terms of achieving the goal. Figure 6 presents the accuracy of the VLM at various levels of differences between the two images. The “difference” between two images is measured as the difference between the ground-truth task progress associated with the images. We discretize the differences into 10 bins along the x axis in Figure 6, where a larger number indicates a greater difference between two images in terms of task progress. On the y axis, the green, orange, and blue bars represent the ratio where the VLM preference label is correct, incorrect, or when there is no preference.

For all tasks, we observe a general trend of increasing accuracy, decreasing uncertainty, and decreasing error as the differences between the images increase, which aligns with intuition. This trend is most clear and consistent for the...
We provide analysis of the accuracy of the VLM preference labels, compared to ground-truth preference labels defined according to the environment’s reward function. The x-axis represents different levels of differences between the image pairs, discretized into 10 bins, where the difference is measured as the difference between the ground-truth task progress associated with the image pairs. The y-axis shows the ratio where the VLM preference labels are correct, incorrect, or when it does not have a preference over the image pairs.

We compare how well the learned reward by RL-VLM-F and VLM Score align with the ground-truth task progress on 3 MetaWorld tasks along an expert trajectory. As shown, RL-VLM-F generates rewards that align better with the ground-truth task progress. The learned rewards are averaged over 3 trained reward models with different seeds, and the shaded region represents the standard error.

CartPole, Open Drawer and Soccer tasks. Overall, for all tasks, we find that the VLM is able to generate more correct preference labels than incorrect ones, and as shown in Figure 4, the accuracy of VLM-generated preference labels is sufficient for learning a good reward function and policy.

6.4. How does the learned reward align with the task progress?

Figure 7 plots the learned rewards (averaged over 3 trained reward models with different random seeds) as well as the true task progress on three MetaWorld environments along an expert trajectory that fully solves the task. Note the ground-truth task progress is not the same as the author-provided reward function: the author provided reward is a shaped version of the task progress. For Open Drawer, the task progress is measured as the distance the drawer has been pulled out; For Soccer, it is measured as the negative distance between the soccer ball and the goal; For Sweep Into, it is measured as the negative distance between the cube and the hole. We normalize both the ground-truth task progress and the learned reward into the range of [0, 1] for a better comparison between them. An ideal learned reward should increase as the time step increases along the expert trajectory, as like the ground-truth task progress. As shown, the reward learned by RL-VLM-F aligns better with the ground-truth task progress compared with the VLM Score baseline. We do notice the learned reward tends to be noisy; but it is sufficient for deriving useful policies for solving the task, as shown by earlier experiments. For Open drawer, we notice that the reward produced by VLM Score remains zero. This is likely because, during training, most of the scores given by the VLM are 0, and the model learns to predict 0 at all time steps to minimize the regression loss. We find the CLIP and BLIP-2 scores on these environments are generally noisy; the corresponding plots can be found in Appendix F.
7. Conclusion and Future Work

In this work, we present RL-VLM-F, a method that automatically generates reward functions via querying VLMs with preferences given a task descriptions and image observations for a wide range of tasks. We demonstrate our proposed method’s effectiveness on rigid, articulated, and deformable object manipulation tasks.

Future work could extend RL-VLM-F to an active learning context, exploring both easy and informative VLM queries for more efficient reward learning. The adaptable nature of our method allows for the integration of more advanced VLMs when they become available, potentially addressing more complex tasks. Additionally, our approach offers a practical pathway to applying RL in real-world settings, where obtaining reward functions is often difficult.

Impact Statement

As we use pre-trained Vision Language Models for generating the reward functions, the bias presented in the VLMs might be inherited into the reward function and subsequently the learned policy. As a result, one might want to examine the behavior of the learned policy before deploying it to safety critical applications. Other than this point, we do not anticipate any societal consequences of our work that must be specifically highlighted here.

References


Appendix

A. Details on Tasks and Environments

We run our method and baselines on CartPole from openAI Gym (Brockman et al., 2016), three rigid and articulated object manipulation tasks from MetaWorld (Yu et al., 2020), and three deformable object manipulation tasks from Soft-Gym (Lin et al., 2021). We describe the observation space and action space for those tasks as follows:

A.1. Observation Space

For policy learning with SAC, we use state-based observations; for reward learning, we use high dimensional RGB image observations, rendered by the simulator. We now detail the state-based observation space for each task.

MetaWorld Tasks. For MetaWorld tasks, we follow the setting in the original paper (Yu et al., 2020). The state observation always has 39 dimensions. It consists of the position and gripper status of the robot’s end-effector, the position and orientation of objects in the scene, and the position of the goal.

CartPole. The state observation has 4 dimensions, including the position and velocity of the cart, as well as the angle and angular velocity of the pole.

Cloth Fold. The state observation is the position of a subset of the particles in the cloth mesh. The cloth is of size 40 x 40, and we uniformly subsample it to be of size 8 x 8. The state is then the position of the picker, and the positions of all those subsampled particles.

Straighten Rope. The state observation is the positions of all particles on the rope and has 36 dimensions.

Pass Water. The state observation includes the size (width, length, height) of the container, the target container position, height of the water in the container, amount of water inside and outside of the container. The state observation has 7 dimensions.

A.2. Action Space

For all environments, we normalize the action space to be within \([-1, 1]\). Below we describe the action space for each environment.

MetaWorld Tasks. For MetaWorld tasks, the action space always has four dimensions. It includes the change in 3D position of the robot’s end-effector followed by a normalized torque that the gripper fingers should apply.

CartPole. The original action space is a discrete value in 0, 1, indicating the direction of the fixed force the cart is pushed with. We modified it to be continuous within range [0, 1] such that SAC can be used as the learning algorithm.

The continuous action represents the force applied to the pole.

Cloth Fold. For this task, we use a pick-and-place action primitive. We assume that the corner of the cloth is grasped when the task is initialized. The action is the 2D target place location.

Straighten Rope. For this task, we use two pickers, one at each end of the rope, to control the rope. Therefore, the action space is the 3D delta positions for each picker and has 6 dimensions in total.

Pass Water. The motion of the glass container is constrained to be in one dimension. Therefore, the action also has a dimension of 1 and is the delta position of the container along the dimension.

B. Hyper-parameters and Network Architectures

B.1. Image-based Reward Learning

For the image-based reward model, we use a 4-layer Convolutional Neural Network for MetaWorld tasks and CartPole and a standard ResNet-18 (He et al., 2016) for the three deformable object manipulation tasks. Following PEBBLE (Lee et al., 2021a), we also use an ensemble of three reward models and use tanh as the activation function for outputting reward. For RL-VLM-F, we train the model by optimizing the cross-entropy loss, defined in Equation 2. For VLM Score, we train the model by optimizing the MSE loss between the predicted score and ground-truth score output by the VLM. For both methods, we use ADAM (Kingma & Ba, 2014) as the optimizer with an initial learning rate of 0.0003.

B.2. Policy Learning

Following PEBBLE (Lee et al., 2021a), we use SAC as the off-policy learning algorithm. We follow the network architectures for the actor and critic and all the hyper-parameter settings in the original paper for policy learning.

B.3. Training details

Our implementation is based on PEBBLE (Lee et al., 2021a). Below we describe the feedback collection schedule for each task. For all tasks, we use a segment size of 1. We summarize the number of queries per feedback session (M in Algorithm 1), the frequency at which we collect feedback in terms of environment steps (K in Algorithm 1), and the maximum budget of queries (N) for each task in Table 1. For Cloth Fold, we have to use a small number of maximum budget of queries due to the quota limitation of GPT-4V.
Table 1. Hyper-parameters for feedback learning schedule.

<table>
<thead>
<tr>
<th>Task</th>
<th>M</th>
<th>K</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Drawer</td>
<td>40</td>
<td>4000</td>
<td>20000</td>
</tr>
<tr>
<td>Soccer</td>
<td>40</td>
<td>4000</td>
<td>20000</td>
</tr>
<tr>
<td>Sweep Into</td>
<td>40</td>
<td>4000</td>
<td>20000</td>
</tr>
<tr>
<td>CartPole</td>
<td>50</td>
<td>5000</td>
<td>10000</td>
</tr>
<tr>
<td>Cloth Fold</td>
<td>50</td>
<td>1000</td>
<td>500</td>
</tr>
<tr>
<td>Straighten Rope</td>
<td>100</td>
<td>5000</td>
<td>12000</td>
</tr>
<tr>
<td>Pass Water</td>
<td>100</td>
<td>5000</td>
<td>12000</td>
</tr>
</tbody>
</table>

Figure 8. On the Fold Cloth task, we find the performance of GPT-4V to be better than Gemini-Pro, possibly due to the complex visual appearance of the cloth.

C. Baselines

C.1. VLM score

For this baseline, we use the same amount of queries (K) at the same frequency (M) as in our method to ask VLM to directly output a score between 0 to 1. The reward model’s architecture is the same as our method, except that the model is trained with regression loss to regress to VLM’s output score instead of classification loss as done in our method.

C.2. RoboCLIP

In RoboCLIP, the backbone video-language model is S3D (Xie et al., 2018), trained on clips of human activities paired with textual descriptions from the HowTo100M dataset (Miech et al., 2019). Given the assumption that the model generalizes to unseen robotic environments, we applied this baseline solely to the three MetaWorld tasks that contain a robot in the scene. We obtain the implementation directly from the authors. To maintain uniform assumptions across methods, we compare against the RoboCLIP variant that only uses a text description instead of a video demonstration to compute the similarity score with the agent’s episode rollout for reward computation. According to the original paper, this text-only variant of RoboCLIP underperforms the video-based method, corroborating the lower performance observed in our tasks.

D. Prompts

D.1. RL-VLM-F and VLM Score

For both RL-VLM-F and VLM Score, we use a unified query template combined with specific task goal descriptions. The templates for RL-VLM-F and VLM Score are shown in Figure 9 and Figure 10:

The only task-specific part in both prompts is the task goal description. We use the same set of descriptions for both methods. We summarize the textual description for each task in Table 2.

D.2. CLIP Score and BLIP-2 Score

The task descriptions for both CLIP Score and BLIP-2 Score baselines are summarized in Table 3. The semantic meaning is almost identical to those used by RL-VLM-F and VLM Score, except that the description is structured differently. For CartPole, we used the exact same prompt as in (Rocamonde et al., 2023), since they reported successful learning of this task using that prompt.

D.3. RoboCLIP

For the task descriptions for the RoboCLIP baseline, we followed the format used in the original paper (Sontakke et al., 2023). We summarize the text descriptions in Table 4.

E. Ablation Study: Influence of Using Different VLMs

For RL-VLM-F and the VLM score baseline, we use Gemini-Pro (Team et al., 2023) as the VLM for all tasks except Fold Cloth. We find Gemini-Pro to perform poorly on Fold Cloth, so we instead use GPT-4V (OpenAI, 2023) as the VLM for this task for both methods. Figure 8 compares the learning performance of Gemini-Pro versus GPT-4V on the task of Fold Cloth. We do observe GPT-4V to achieve much better performance on this task than Gemini-Pro. The poorer performance of Gemini-Pro on this task could be possibly due to the more complex visual reasoning required for deformable cloth.

F. More Visualization of the Learned Reward

Here we show the learned reward from RL-VLM-F and the VLM Score baseline, as well as the CLIP and BLIP-2 score along an expert trajectory on three MetaWorld tasks. We compare the learned reward from RL-VLM-F and the VLM Score / CLIP and BLIP-2 score to the ground-truth task progress. The results are shown in Figure 11. For all three tasks, the reward learned by RL-VLM-F aligns the best with the ground-truth task progress.
Prompt Template for RL-VLM-F (ours)

**Analysis Template**
Consider the following two images:
- Image 1: [Image 1]
- Image 2: [Image 2]
1. What is shown in Image 1?
2. What is shown in Image 2?
3. The goal is to [task description]. Is there any difference between Image 1 and Image 2 in terms of achieving the goal?

**Labeling Template**
Based on the text below to the questions:
[Repeat the 3 questions in the Analysis Template]
[VLM response]
Is the goal better achieved in Image 1 or Image 2? Reply a single line of 0 if the goal is better achieved in Image 1, or 1 if it is better achieved in Image 2.
Reply -1 if the text is unsure or there is no difference.

*Figure 9. Prompt Template for RL-VLM-F.*

Prompt Template for VLM Score

**Analysis Template** Consider the following image:
[Image]
1. What is shown in the image?
2. The goal is [task description]. On a scale of 0 to 1, the score is 1 if the goal is achieved. What score would you give the image in terms of achieving the goal?

**Labeling Template**
Based on the text below to the questions:
[Repeat the 3 questions in the Analysis Template]
[VLM response]
Please reply a single line of the score the text has given. Reply -1 if the text is unsure.

*Figure 10. Prompt Template for VLM Score.*
<table>
<thead>
<tr>
<th>Task Name</th>
<th>Goal Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Open Drawer</strong></td>
<td>to open the drawer</td>
</tr>
<tr>
<td><strong>Soccer</strong></td>
<td>to move the soccer ball into the goal</td>
</tr>
<tr>
<td><strong>Sweep Into</strong></td>
<td>to minimize the distance between the green cube and the hole</td>
</tr>
<tr>
<td><strong>CartPole</strong></td>
<td>to balance the brown pole on the black cart to be upright</td>
</tr>
<tr>
<td><strong>Cloth Fold</strong></td>
<td>to fold the cloth diagonally from top left corner to bottom right corner</td>
</tr>
<tr>
<td><strong>Straighten Rope</strong></td>
<td>to straighten the blue rope</td>
</tr>
<tr>
<td><strong>Pass Water</strong></td>
<td>to move the container, which holds water, to be as close to the red circle as possible without causing too many water droplets to spill</td>
</tr>
</tbody>
</table>

*Table 2. Goal description used in RL-VLM-F and VLM Score baseline.*

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Goal Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Open Drawer</strong></td>
<td>The drawer is opened.</td>
</tr>
<tr>
<td><strong>Soccer</strong></td>
<td>The soccer ball is in the goal.</td>
</tr>
<tr>
<td><strong>Sweep Into</strong></td>
<td>The green cube is in the hole.</td>
</tr>
<tr>
<td><strong>CartPole</strong></td>
<td>pole vertically upright on top of the cart.</td>
</tr>
<tr>
<td><strong>Cloth Fold</strong></td>
<td>The cloth is folded diagonally from top left corner to bottom right corner.</td>
</tr>
<tr>
<td><strong>Straighten Rope</strong></td>
<td>The blue rope is straightened.</td>
</tr>
<tr>
<td><strong>Pass Water</strong></td>
<td>The container, which holds water, is as close to the red circle as possible without causing too many water droplets to spill</td>
</tr>
</tbody>
</table>

*Table 3. Goal description used in CLIP Score and BLIP-2 Score.*

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Goal Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Open Drawer</strong></td>
<td>robot opening green drawer</td>
</tr>
<tr>
<td><strong>Soccer</strong></td>
<td>robot pushing the soccer ball into the goal</td>
</tr>
<tr>
<td><strong>Sweep Into</strong></td>
<td>robot sweeping the green cube into the hole on the table</td>
</tr>
</tbody>
</table>

*Table 4. Goal description used in RoboCLIP.*
Figure 11. Comparison of learned reward functions from RL-VLM-F and VLM Score, as well as CLIP and BLIP-2 score to the ground-truth task progress along a trajectory rollout on three MetaWorld tasks. From left column to right: reward learned by RL-VLM-F, reward learned by VLM Score, CLIP Score, BLIP-2 Score. From top row to bottom: Open Drawer, Soccer, and Sweep Into. The reward learned by RL-VLM-F aligns the best across all compared methods.