GoalLadder: Incremental Goal Discovery with Vision-Language Models

Alexey Zakharov
University of Oxford
alexey.zakharov@cs.ox.ac.uk

Shimon Whiteson
University of Oxford
shimon.whiteson@cs.ox.ac.uk

Abstract

Natural language can offer a concise and human-interpretable means of specifying reinforcement learning (RL) tasks. The ability to extract rewards from a language instruction can enable the development of robotic systems that can learn from human guidance; however, it remains a challenging problem, especially in visual environments. Existing approaches that employ large, pretrained language models either rely on non-visual environment representations, require prohibitively large amounts of feedback, or generate noisy, ill-shaped reward functions. In this paper, we propose a novel method, **GoalLadder**, that leverages vision-language models (VLMs) to train RL agents *from a single language instruction in visual environments*. GoalLadder works by incrementally discovering states that bring the agent closer to completing a task specified in natural language. To do so, it queries a VLM to identify states that represent an improvement in agent's task progress and to rank them using pairwise comparisons. Unlike prior work, GoalLadder does not trust VLM's feedback completely; instead, it uses it to rank potential goal states using an ELO-based rating system, thus reducing the detrimental effects of noisy VLM feedback. Over the course of training, the agent is tasked with minimising the distance to the top-ranked goal in a learned embedding space, which is trained on unlabelled visual data. This key feature allows us to bypass the need for abundant and accurate feedback typically required to train a well-shaped reward function. We demonstrate that GoalLadder outperforms existing related methods on classic control and robotic manipulation environments with the average final success rate of $\sim$95% compared to only $\sim$45% of the best competitor.

1 Introduction

Reinforcement learning (RL) relies critically on effective reward functions to guide agents toward desired behaviours. However, crafting reward functions by hand requires significant human labour and domain expertise. Fortunately, many RL tasks can be succinctly described in natural language (e.g., 'open the drawer', 'close the window'), from which humans can understand the task at hand and approximate a reward function to reach the goal state. In particular, extracting a reward function from a language instruction requires understanding the semantics of the task description, associating its components with the given environment, and integrating this information with prior, common-sense knowledge. The rise of large language models (LLMs) brings hope of automating this process – given a language instruction, can we automatically extract an effective reward function?

Indeed, LLMs can perform well on open-ended question answering, code generation, and be vast reservoirs of common-sense knowledge [1, 2, 3, 4]. Moreover, multimodal models have even broader applicability to tasks involving images or other modalities [5, 6, 7, 8, 9, 10]. These properties make pretrained language models potentially suitable for providing learning signals to reinforcement learning algorithms, such as by generating preference-based feedback [11] or outputting a reward function directly [12].

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Prior work attempts to use LLMs to generate reward functions given access to environment code or interpretable state representations [13, 14, 12], limiting their applicability. Another line of work uses vision-language models (VLMs) to define or learn a reward function given a language instruction and visual observations. In particular, two broad strategies have emerged: (i) embedding-based [2, 15, 16, 17], which aligns visual observations with a language instruction in the embedding space of a VLM; and (ii) preference-based [11], which prompts a VLM to rank segments of an agent's behaviour by how well they match a textual specification, and trains a reward function from the collected preference data.

Both approaches present certain challenges. Embedding-based methods employing CLIP [5] tend to yield noisy reward functions due to the mismatch between the CLIP training data and the observations from the tested environments [16]. Similarly, these methods require that every observation is embedded to produce rewards, making them expensive and inefficient in using large pretrained models. Preference-based methods like RL-VLM-F [11] produce reward functions with less noise and better correlation with task progress but are nevertheless substantially noisy due to mistakes made by a VLM when comparing trajectories. Erroneous feedback plagues the preference label dataset (at an unknown frequency) making it challenging to robustly train a well-shaped reward function without overfitting to mislabelled samples. Furthermore, although preference-based approaches are more sample-efficient than embedding-based ones, they nevertheless require many VLM queries to train a reward function that generalises well.

In this paper, we argue that, to be effective in practice, approaches employing VLMs for feedback must address two key issues: (a) robustness to noisy VLM feedback; and (b) query efficiency in using VLMs for feedback, given that repeated large-scale queries to VLMs can be prohibitively expensive.

To this end, we propose **GoalLadder**, an algorithm designed to address both of these challenges. Our method leverages a VLM to incrementally discover environment states that bring the agent closer to completing a task. To avoid the problems of prior methods, GoalLadder performs repeated comparisons of a small subset of visual observations and maintains a persistent, ELO-based utility measure (*rating*) over states. This process allows GoalLadder to progressively refine its estimates of state utilities without being severely affected by noisy VLM feedback. Furthermore, compared to previous methods, GoalLadder requires substantially fewer VLM queries to learn desired behaviours, since rewards are defined as distances between visual observations in a learned embedding space that is trained on unlabelled data, allowing for reward generalisation to unseen states without explicit feedback. Using two classic control and five robotic manipulation tasks, we demonstrate that our method significantly outperforms prior work that utilises vision-language models for RL, with the average final success rate of $\sim$95% compared to only $\sim$45% of the best competitor. GoalLadder exhibits impressive performance even against the oracle agent that has access to ground-truth reward – nearly matching it across all tested tasks and convincingly surpassing it on one.

2 Related Work

Vision-language models Recent advances in large language models [18, 19, 20] demonstrate impressive abilities for reasoning and common sense in text and other modalities such as vision [5, 6, 7, 8, 9, 10]. Vision-language models are trained on a joint visual and textual representation space, allowing the capabilities of LLMs to be applied within the visual domain. Nevertheless, existing VLMs suffer from poor spatial understanding [21], making it difficult to reliably apply them to spatial reasoning tasks. Our approach takes this into account by design, using a robust pairwise evaluation strategy to reduce the effects of erroneous feedback on spatial tasks.

Language models in RL The use of language models in reinforcement learning has been predominantly about making LLMs write code, particularly to design a reward function [13, 14, 12]. Although these approaches may improve as language models get better at code comprehension, it is unclear how they would perform in arbitrarily complex physical simulations or transfer to real-world environments. LLMs have also been used to provide a reward signal in text-based environments [22, 23].

Vision-language models in RL Several works explore the integration of VLMs in RL. One avenue is using the embeddings of the VLMs directly in the definition of the reward. For example, [16] use the CLIP model [5] to embed the task description specified in text, as well as an environment's visual

observations, and define the reward as the cosine similarity between the image and text embedding. However, the resulting reward functions tend to be noisy, which is often attributed to the mismatch between the training data of the VLMs and the environments they are applied to [16]. To this end, Baumli et al. [17] fine-tune a CLIP model with contrastive learning to output the success of an agent in reaching a text-based goal. Other papers use vision-language models to *align* visual observations with language descriptions of a task [2, 24, 15, 25], to perform incremental trajectory improvement from human feedback [26] or to elicit better task-relevant representations [27].

VLMs can also be used for direct feedback to learn a reward function from VLM goal-conditioned preferences [11]. In particular, Wang et al. [11] use VLMs to provide preference labels over states of visual observations conditioned on a task description and use these labels to learn a reward function with Reinforcement Learning from Human Preferences (RLHF) [28]. Although this work improves upon previous methods, noise in the learned reward function remains evident. Furthermore, for the experiments with robotic manipulation, the authors remove the robot from the image to improve the VLM's ability to identify goal states in object-oriented tasks. This trick, however, also reduces the amount of information a VLM can use to identify incrementally better states. In contrast, GoalLadder solves the tasks without environment modifications and utilises an ELO-based rating system to reduce the effects of noisy VLM feedback.

3 GoalLadder

3.1 Preliminaries

Reinforcement learning We formulate the problem as a Markov Decision Process (MDP), defined as a tuple $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma\}$. Here, $\mathcal{S}$ represents the state space of the environment, $\mathcal{A}$ is the agent's action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ defines the transition dynamics, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to [0, \infty)$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. At each timestep t, an agent takes an action a_t, transitions to state s_{t+1}, and receives reward r_t. In our setup, we assume that the agent similarly receives an image observation o_t that represents the state of the environment. The goal of the agent is to maximise the sum of discounted rewards in the environment, defined as $G = \sum_{k=0}^{\infty} \gamma^k r(s_k, a_k)$.

Soft-Actor Critic We use soft-actor critic (SAC) [29] as the RL backbone of our agent. However, in GoalLadder, the reward function changes periodically as new targets are being set over the course of training. The vanilla implementation of SAC can struggle with non-stationary reward given that it reuses past data from the replay buffer, which may not be up to date with respect to the latest reward function. As such, similar to [30], we periodically relabel all of the agent's trajectories with the latest reward function.

Assumptions We assume access to a language instruction, l, that succinctly describes the task of the agent in any given environment (e.g., "open the drawer"). Furthermore, we consider tasks whose goal can be represented by a visual observation (not necessarily unique).

3.2 Method

Overview We propose GoalLadder, which uses a vision-language model to incrementally discover states that bring the agent closer to completing a task specified in natural language. GoalLadder uses a VLM in two ways: (i) to identify *candidate* goals, denoted as g, to be put into a ranking buffer $\mathcal{B}_g = \{g_1, g_2, ...\}$ for further evaluation, and (ii) to rate the candidate goals, with rating denoted as e, based on their proximity to the goal state specified in language. The candidate goals are discovered and ranked *online*, as the RL agent trains and collects new observations according to its latest SAC policy. Over the course of training, the top-ranked goal serves as the target state that the RL agent is tasked with achieving.

GoalLadder involves several stages that are expanded upon in Sections 3.2.1-3.2.3:

1. *Collection*: the RL agent collects an episode by interacting with the environment according to its current SAC policy.
2. *Discovery*: a VLM is queried to determine whether any of the collected observations represent an improvement over the current top-rated candidate goal.

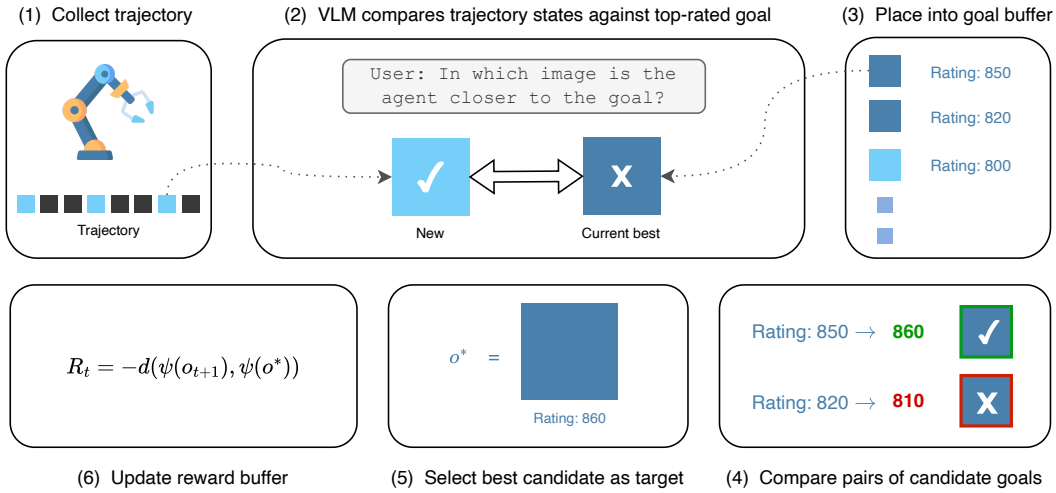

Figure 1: **Summary of GoalLadder. (1-2)** An agent collects a trajectory using the current SAC policy. Trajectory observations are uniformly sampled and compared against the current best candidate goal in the buffer. **(3)** If successful, they are placed into the candidate goal buffer. **(4)** Pairs of candidate goals are sampled and compared against each other using a VLM. Their ratings are updated after each comparison. **(5-6)** Periodically, the target state is updated with the current top-rated goal from the candidate buffer. Reward function is defined as the negative distance to the current best candidate goal in a learned embedding space; the RL agent is then trained with this reward. The process returns to step 1 and repeats.

3. *Ranking*: a VLM is queried with sampled pairs of existing candidate goals from the buffer for comparison and their ratings are updated accordingly.
4. *Training*: The agent is trained to minimise the distance to the top-ranked candidate goal in a learned embedding space.

Together, the stages represent a repeating process, in which the RL agent trains and collects new observations, while candidate goals are being discovered and ranked. A visual illustration of GoalLadder operations is provided in Figure 1.

3.2.1 Discovery of candidate goals

To succeed, GoalLadder must address the question: *Which states are worth evaluating as candidate goal states?* Since our capacity to query the VLM is limited, we need to filter out irrelevant states before they are entered into the goal buffer for detailed evaluation. This ensures that the VLM focuses on the most promising states, avoids wasting compute on obvious or trivial cases, and more quickly converges on the final goal. To this end, we maintain a buffer of candidate goals, $\mathcal{B}_g$, where each candidate goal g_i consists of an image o_i representing the state and the goal's rating e_i, such that $g_i = (o_i, e_i)$. Given the current top-rated candidate goal,

$$g^* = \arg\max_{g_i \in \mathcal{B}_g} e_i, \quad \text{where } g^* = (o^*, e^*),$$

and a randomly sampled state image o_j from the agent's latest collected trajectory, the VLM is asked to decide which of the two images, o^* or o_j, represents a state that is closer to the language-specified task goal, l. We denote the VLM output by

$$y = \text{VLM}(o^*, o_j, l), \quad y \in \{-1, 0, 1\},$$

where $y = -1$ indicates that there is no decision or the images are equivalent, $y = 0$ means g^* is deemed a better goal than o_j, and $y = 1$ means o_j is deemed a better goal than g^*.

If $y = 1$, the newly sampled state o_j is inserted into $\mathcal{B}_g$ as a fresh candidate goal with a standard initial rating. Conversely, if $y = 0$ or $y = -1$, o_j is disregarded in order to keep the goal buffer focused on higher-quality candidates.

3.2.2 Rating of candidate goals

Once the goal buffer is large enough, GoalLadder begins querying a VLM to compare the candidate goal states and get a better estimate of their true rating. Specifically, a pair of candidate goals is sampled, $(g_i, g_j) \sim \mathcal{B}_g$, and the same querying process, as in Section 3.2.1, is used.

Ideally, our goal ratings would have two main properties: (1) be robust to noisy VLM outputs and (2) allow for quick updates to a candidate goal's rating when new and better goals are discovered. In GoalLadder, we draw inspiration from the ELO chess rating system [31] to maintain a robust, adaptive measure of each candidate goal's utility. In chess, players' ratings are updated after every match based on two factors: the observed outcome (win, lose, or draw) and the expected outcome given the players' current ratings.

In ELO, the expected score for candidate goal g_i against g_j is given by the logistic function:

$$E_i = \frac{1}{1 + 10^{(e_j - e_i)/C}},$$

where e_i and e_j are the current ratings of g_i and g_j, respectively, and C is a constant controlling how sensitive the expected scores are to rating differences.

Given an observed result $S_i \in \{-1, 1, 0\}$ ("win," "loss," or "draw"), we update the rating of g_i as:

$$e_i \leftarrow e_i + T\big(S_i - E_i\big),$$

and similarly for g_j. Here, T is a constant that governs how quickly ratings are adjusted.

The ELO rating system incrementally absorbs noisy VLM comparisons, adaptively adjusts ratings when new evidence shows a goal is better or worse, and continuously refines its estimates to converge on a stable hierarchy of candidate goals aligned with the language-specified target.

3.2.3 Defining the reward function

Once a candidate goal $g^* \in \mathcal{B}_g$ emerges as the highest-rated goal, we treat its image o^* as the agent's current best guess for the true task objective. As such, we wish to encourage the agent to bring its state as close as possible to o^* in an appropriate metric space. However, defining a distance directly in an environment's intrinsic state-space may be unreliable given environment-to-environment differences.

To address this, GoalLadder learns a visual feature extractor to produce a compact latent representation of environment observations. We achieve this using a variational autoencoder training objective [32],

$$\mathcal{L} = -\mathbb{E}_{\psi(z_t|o_t)} \log p_\theta(o_t \mid z_t) + D_{\mathrm{KL}}\left(\psi(z_t \mid o_t) \,\|\, p(z_t)\right), \tag{1}$$

where the encoder $\psi(\cdot)$ is trained to produce a latent vector z_t representing observation o_t. We implement the feature extractor using a simple convolutional neural network architecture, the details of which can be found in Appendix A.

By training on a diverse set of observations the agent encounters, we obtain a latent space that captures salient visual (and potentially semantic) features of the environment. Finally, we define the agent's reward at time step $t-1$ to be:

$$R(s_{t-1}, a_{t-1}) = -\,d\big(z_t, z^*\big),$$

where $z_t = \psi(o_t)$, $z^* = \psi(o^*)$, and $d(\cdot, \cdot)$ is the Euclidean distance. As such, the agent is incentivised to minimise the distance to the top-rated goal g^*.

As the candidate goals are being discovered and rated during training, we periodically update our reward function and relabel all stored transitions [30] with respect to the top-rated candidate goal. Periodicity ensures that sudden or erroneous changes in the top-rated goal do not significantly destabilise the training process, while also keeping the best goal estimate up-to-date.

3.2.4 Implementation details

We use Gemini 2.0 Flash[1] as our VLM backbone. The top-rated candidate goal is selected as the new target every $L = 5000$ environment steps. The goal buffer size is capped at $|\mathcal{B}_g| = 10$, as the lowest

[1] `https://ai.google.dev/gemini-api/docs/models#gemini-2.0-flash`

Algorithm 1: GOALLADDER: Pseudo algorithm

Input : Task description l, experience buffer $\mathcal{D}$, SAC, VLM, reward update schedule L

Initialise candidate goal buffer $\mathcal{B}_g$ with random observations
while *training* **do**
  Collect episode $\mathcal{E} \sim \pi$ and store in $\mathcal{D}$
  `// Discovery and Ranking`
  if $t \bmod K = 0$ **then**
    Sample M trajectory observations $\{o_1, \ldots, o_M\} \sim \mathcal{E}$
    Compare each $o_i \in \{o_1, \ldots, o_M\}$ against top-rated candidate using VLM and l
    Update goal buffer $\mathcal{B}_g$ based on the feedback
    Compare M pairs of $(g_i, g_j) \sim \mathcal{B}_g$ using VLM and l
    Update candidate goal ratings, (e_i, e_j)
  end
  `// Update reward function`
  if $t \bmod L = 0$ **then**
    Select the top-rated goal from the buffer, $g^* = \arg\max_{(o_i, e_i) \in \mathcal{B}_g} e_i$
    Update rewards in experience buffer $\mathcal{D}$ using $\|\psi(o) - \psi(o^*)\|_2$
  end
  `// Train`
  if $t \bmod 1 = 0$ **then**
    Update SAC
    Update ψ using a minibatch of observations from $\mathcal{D}$
  end
end

rated candidate goals are removed every L steps. This ensures the comparisons remain focused on the most promising states. VLM queries are performed every K environment steps with M queries per feedback session (see Section 4). SAC gradient update is performed after every environment step. Finally, the VLM prompts are standardised using a single template to ensure consistency and ease of use . Algorithm 1 shows the stepwise, high-level operations of GoalLadder. See Appendix A for further implementation details, including hyperparameters, architectural details, and used computational resources.

4 Experiments

We analyse the performance of our algorithm in the scope of continuous control tasks, from classic control to more complex robotic manipulation. Specifically, we aim to answer the following questions:

1. Can GoalLadder effectively *discover* the best goal over the course of training?
2. Can GoalLadder *solve* continuous-control tasks?
3. Is GoalLadder more *sample-efficient* with respect to VLM queries than related methods?

Environments We run a variety of continuous-control experiments to investigate GoalLadder's performance. In particular, we use two classic control environments (*CartPole, MountainCar*) from OpenAI Gym [33] and five robotic manipulation environments (*Drawer Close, Drawer Open, Sweep Into, Window Open, Button Press*) from the Metaworld suite [34]. Importantly, unlike prior works [16, 11], we do not perform any special environment modifications to help VLMs discern task progress. We use feedback rates of $K = 2000$ with $M = 5$ for OpenAI Gym environments and $K = 500$ with $M = 5$ for the Metaworld environments.

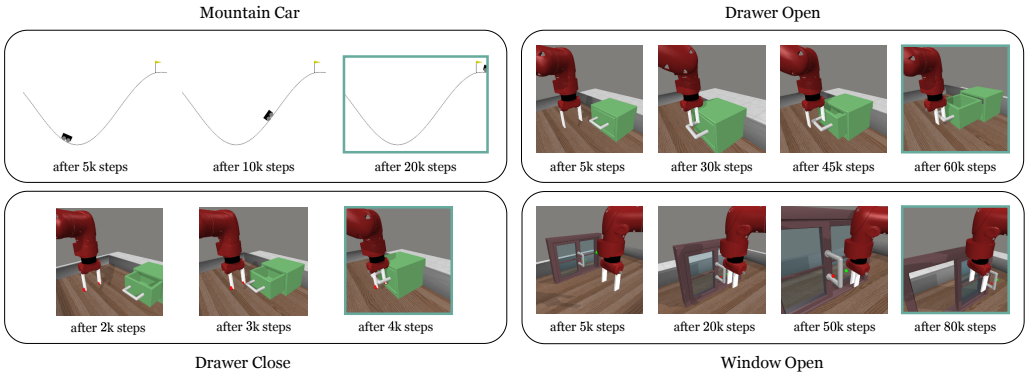

Figure 2: **Top-rated candidate goals during training.** Top-rated candidate goals in the Mountain Car environment follow a natural progression of the car getting closer to the top of the hill. Similarly, in the Metaworld tasks, GoalLadder rapidly discovers states that bring the robotic arm closer to the true task objective, until the best goal is discovered.

Baselines We compare GoalLadder to the following baselines:

- **Ground-truth reward (Oracle)**: A strong baseline in which the RL algorithm is trained with the environment's intrinsic reward. This benchmark acts as the oracle and should serve as an upper bound on the performance of our method.
- **VLM-RM** [16]: Uses CLIP [5] to embed the task description and an image observation and defines the reward as the cosine-similarity between the two embeddings.
- **RoboCLIP** [15]: Employs a pre-trained video-language model, S3D [35], to compute rewards based on the similarity between an agent's video trajectory and either a reference video or a text description. To ensure a fair comparison, we use the text version of RoboCLIP.
- **RL-VLM-F** [11]. Uses a VLM to compare pairs of images with respect to a language instruction. Unlike GoalLadder, RL-VLM-F utilises VLM feedback to train a reward function from the collected labels. For this baseline, we apply *the same feedback rate* used in GoalLadder and use the same vision-language model as backbone – Gemini 2.0 Flash.

4.1 Can GoalLadder discover the best goal?

Figure 2 shows the evolution of the top-rated candidate goals in the buffer over the course of the training for the *MountainCar* and *Metaworld* environments. Despite the VLM's tendency to make mistakes in its comparisons, the best goal consistently rises to the top of the ranking. Based on our experiments, we confirm that GoalLadder can effectively discover the target goal given the language instruction for all tested environments.

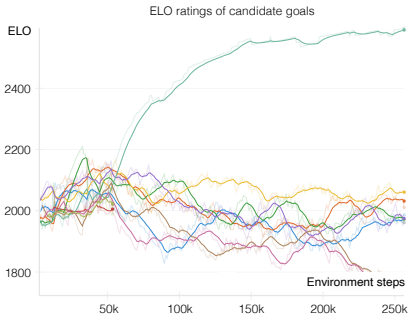

Figure 3: **Candidate goal ratings over time in *Window Open*.** Different colours indicate different candidate goals present in the buffer. GoalLadder quickly singles out the best goal once it is discovered.

Similarly, Figure 3 shows the evolution of the candidate goal ratings in the buffer for the *Metaworld Window Open* environment. During the initial discovery stage (until 50k steps), there is no clear winner and the ratings tend to be similar. Upon the discovery of a clear winner (at around 50k steps), GoalLadder quickly singles it out as the best goal for the agent to pursue. Figure 5 provides further insight by visualising the entire goal buffer over the course of training for the *Drawer Open* task. Two trends can be observed: (i) the buffer is approximately ordered by how close the agent is to achieving the task in each image, and (ii) the buffer becomes progressively filled with better states as training goes on. We observe similar behaviour for all other environments.

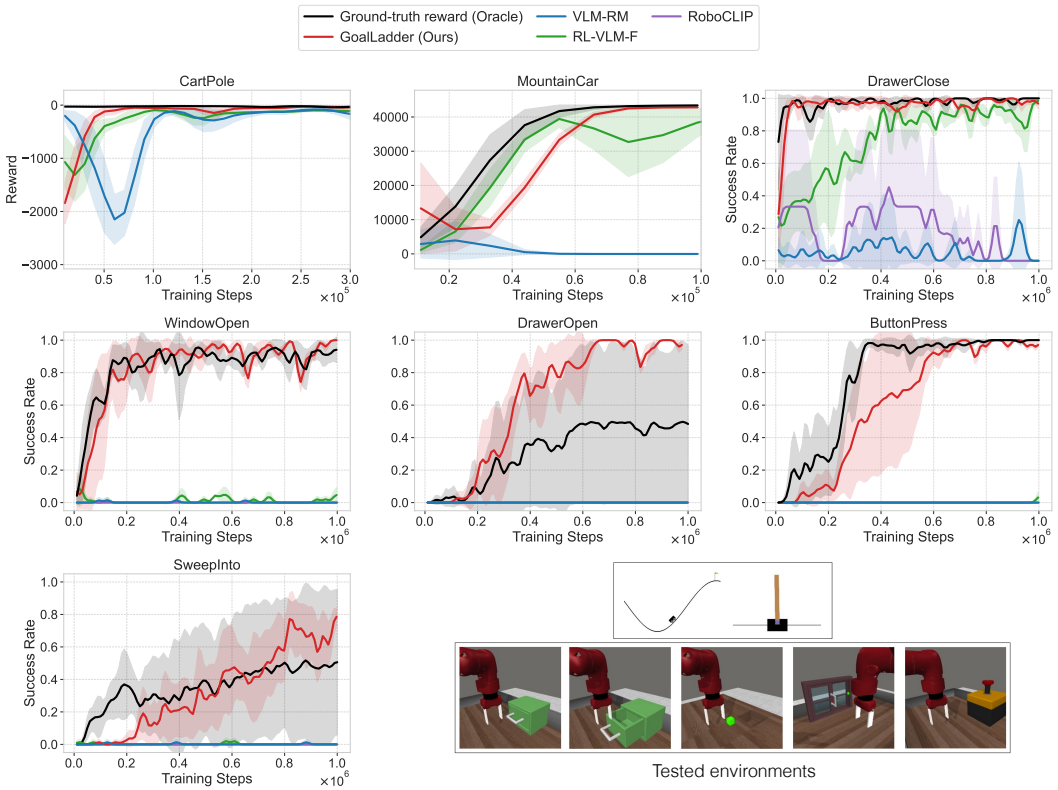

Figure 4: GoalLadder performance against baselines on OpenAI Gym and Metaworld environments. Shaded regions represent the standard deviation over 3 seeds. Averaged across all tasks, GoalLadder achieves a mean success rate of ∼95%, compared to the next best competitor performance of ∼45% achieved by RL-VLM-F.

4.2 Can GoalLadder solve the tasks?

Figure 4 shows that GoalLadder can effectively solve continuous control tasks given only a language instruction, reaching a mean success rate of ∼95% averaged across all tasks. Curiously, our method nearly matches the performance of the ground-truth reward baseline and even convincingly surpasses it for the *Drawer Open* task. Empirically, we find that sometimes the oracle agent learns to reach the drawer, but does not learn to pull the handle to open it. We believe this once again highlights the difficulty of manually designing effective reward functions – an issue that was also previously discussed in Wang et al. [11].

The preference-based baseline, RL-VLM-F, manages to solve the relatively easy tasks – *Cartpole* and *MountainCar* from the OpenAI Gym suite, as well as *Drawer Close* from Metaworld. However, it begins to struggle with the more complicated Metaworld tasks. We attribute this poor performance of RL-VLM-F to the fact that it requires more feedback to learn an effective reward function, as well as ways to mitigate the effects of noisy labels in the preference data. Similarly, VLM-RM and RoboCLIP struggle to learn in most of the tasks, which is in line with previously observed limitations of these methods [11].

4.3 How sample-efficient is GoalLadder?

Using large pretrained models is expensive, so methods that rely on them for feedback must do so efficiently. While embedding-based approaches, like VLM-RM [16], use all environment observations for reward calculation, preference-based methods, like RL-VLM-F [11], can offer better sample efficiency by training a reward function from a limited amount of feedback, in the hope that the learned reward function generalises to unseen states. By contrast, GoalLadder produces reward signals that generalise to unseen states by defining the reward in terms of distances in an embedding space (Section 3.2.3) trained with unsupervised learning. Utilising the *full* extent of agent's experiences, rather than just the labelled observations, for reward definition allows GoalLadder to approximate state utility without collecting large amounts of feedback. Averaged across all *Metaworld* tasks, GoalLadder

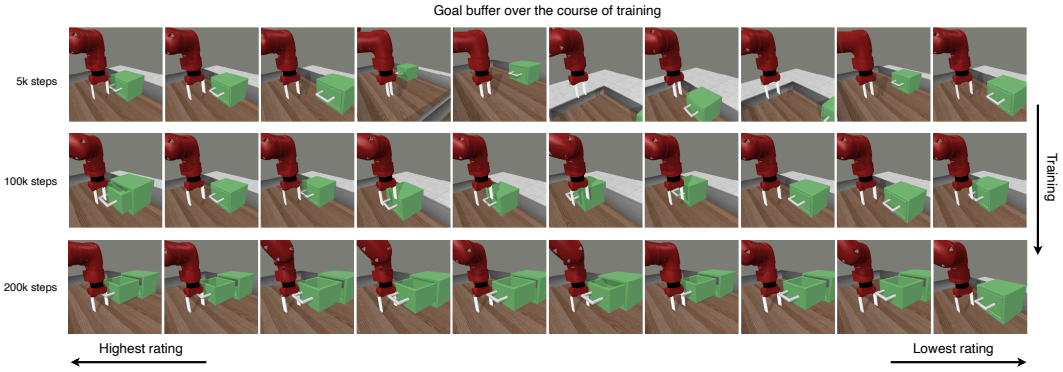

Figure 5: Visualisation of the goal buffer in GoalLadder over the course of training. Each line demonstrates the candidate goals present in the buffer after an indicated number of environment steps. The candidate goals are ordered by ELO rating, from left to right.

learns to solve them after only $\sim$4500 queries. For comparison, PEBBLE [30], a preference-based method that uses *ground-truth reward* preferences, achieves the same performance after attaining an average of $\sim$15000 preference labels on the same tasks. Indeed, we observe that GoalLadder boasts superior sample efficiency compared to the tested baselines. Performance curves in Figure 4 illustrate that RL-VLM-F struggles to learn the tasks using the feedback rate of GoalLadder, while VLM-RM, that embeds all collected observations, fails almost entirely.

5 Discussion

5.1 Limitations

As mentioned in Section 3.2, we assume that the progress or success of a task can be identified from a single image, limiting the application of our method to environments with static goals. Nevertheless, we believe that future work could extend GoalLadder to video-based settings. GoalLadder also largely relies on visual feature similarity for reward definition. In some settings, visual similarity between observations could be a limiting proxy for the underlying state similarity or task progress. Here, more advanced visual embedding techniques could be used instead. Lastly, while our method could always benefit from evaluation on more environments, the costs of using VLMs limit what is feasible.

5.2 Broader impacts

Reinforcement learning from language instructions opens up the door for human-interpretable ways to interact with robotic systems in the real world. At the same time, robotic systems trained with the assistance of large pretrained language models have the potential to inherit any existing biases of these models. Though the scope of this paper is limited to object-oriented environments, we would caution against the application of such models in settings with real humans, until this potential issue is thoroughly investigated.

5.3 Conclusion

We introduced GoalLadder, a method for training reinforcement learning agents from vision-language model feedback using a single language instruction. GoalLadder systematically discovers new and better states that take the agent closer to the final task goal over the course of training. Our method demonstrated impressive performance gains against prior work, while nearly matching the performance of the oracle agent with access to ground-truth reward. We argued that implementing a more comprehensive pairwise comparison technique and using a learned embedding space for reward definition allowed GoalLadder to be robust to noisy feedback and significantly more sample-efficient compared to prior methods.

GoalLadder offers several interesting directions of future research. Firstly, we believe that our method could be extended to video-based settings, depending on the capabilities of the existing vision-language models for video comprehension. Secondly, we believe GoalLadder would benefit from more advanced visual feature extraction techniques or from building a more meaningful latent space, for example using self-supervised learning.

References

[1] Qiyuan He, Yizhong Wang, and Wenya Wang. Can Language Models Act as Knowledge Bases at Scale?, February 2024. URL http://arxiv.org/abs/2402.14273. arXiv:2402.14273 [cs].

[2] Yuchen Cui, Scott Niekum, Abhinav Gupta, Vikash Kumar, and Aravind Rajeswaran. Can Foundation Models Perform Zero-Shot Task Specification For Robot Manipulation? In *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*, pages 893–905. PMLR, May 2022. URL https://proceedings.mlr.press/v168/cui22a.html. ISSN: 2640-3498.

[3] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. Language Models as Knowledge Bases?, September 2019. URL http://arxiv.org/abs/1909.01066. arXiv:1909.01066 [cs].

[4] Anastasia Kritharoula, Maria Lymperaiou, and Giorgos Stamou. Language Models as Knowledge Bases for Visual Word Sense Disambiguation, October 2023. URL http://arxiv.org/abs/2310.01960. arXiv:2310.01960 [cs].

[5] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning*, pages 8748–8763. PMLR, July 2021. URL https://proceedings.mlr.press/v139/radford21a.html. ISSN: 2640-3498.

[6] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision. In *Proceedings of the 38th International Conference on Machine Learning*, pages 4904–4916. PMLR, July 2021. URL https://proceedings.mlr.press/v139/jia21b.html. ISSN: 2640-3498.

[7] Lewei Yao, Runhui Huang, Lu Hou, Guansong Lu, Minzhe Niu, Hang Xu, Xiaodan Liang, Zhenguo Li, Xin Jiang, and Chunjing Xu. FILIP: Fine-grained Interactive Language-Image Pre-Training, November 2021. URL http://arxiv.org/abs/2111.07783. arXiv:2111.07783 [cs].

[8] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to Prompt for Vision-Language Models. *International Journal of Computer Vision*, 130(9):2337–2348, September 2022. ISSN 1573-1405. doi: 10.1007/s11263-022-01653-1. URL https://doi.org/10.1007/s11263-022-01653-1.

[9] Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Qiao. CLIP-Adapter: Better Vision-Language Models with Feature Adapters. *International Journal of Computer Vision*, 132(2):581–595, February 2024. ISSN 1573-1405. doi: 10.1007/s11263-023-01891-x. URL https://doi.org/10.1007/s11263-023-01891-x.

[10] Florian Bordes, Richard Yuanzhe Pang, Anurag Ajay, Alexander C. Li, Adrien Bardes, Suzanne Petryk, Oscar Mañas, Zhiqiu Lin, Anas Mahmoud, Bargav Jayaraman, Mark Ibrahim, Melissa Hall, Yunyang Xiong, Jonathan Lebensold, Candace Ross, Srihari Jayakumar, Chuan Guo, Diane Bouchacourt, Haider Al-Tahan, Karthik Padthe, Vasu Sharma, Hu Xu, Xiaoqing Ellen Tan, Megan Richards, Samuel Lavoie, Pietro Astolfi, Reyhane Askari Hemmat, Jun Chen, Kushal Tirumala, Rim Assouel, Mazda Moayeri, Arjang Talattof, Kamalika Chaudhuri, Zechun Liu, Xilun Chen, Quentin Garrido, Karen Ullrich, Aishwarya Agrawal, Kate Saenko, Asli Celikyilmaz, and Vikas Chandra. An Introduction to Vision-Language Modeling, May 2024. URL http://arxiv.org/abs/2405.17247. arXiv:2405.17247 [cs].

[11] Yufei Wang, Zhanyi Sun, Jesse Zhang, Zhou Xian, Erdem Biyik, David Held, and Zackory Erickson. RL-VLM-F: Reinforcement Learning from Vision Language Foundation Model Feedback, March 2024. URL http://arxiv.org/abs/2402.03681. arXiv:2402.03681 [cs].

[12] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-Level Reward Design via Coding Large Language Models, April 2024. URL http://arxiv.org/abs/2310.12931. arXiv:2310.12931 [cs].

[13] Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, Brian Ichter, Ted Xiao, Peng Xu, Andy Zeng, Tingnan Zhang, Nicolas Heess, Dorsa Sadigh, Jie Tan, Yuval Tassa, and Fei Xia. Language to Rewards for Robotic Skill Synthesis, June 2023. URL http://arxiv.org/abs/2306.08647. arXiv:2306.08647 [cs].

[14] Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. Text2Reward: Reward Shaping with Language Models for Reinforcement Learning, May 2024. URL http://arxiv.org/abs/2309.11489. arXiv:2309.11489 [cs].

[15] Sumedh A. Sontakke, Jesse Zhang, Sébastien M. R. Arnold, Karl Pertsch, Erdem Bıyık, Dorsa Sadigh, Chelsea Finn, and Laurent Itti. RoboCLIP: One Demonstration is Enough to Learn Robot Policies, October 2023. URL http://arxiv.org/abs/2310.07899. arXiv:2310.07899 [cs].

[16] Juan Rocamonde, Victoriano Montesinos, Elvis Nava, Ethan Perez, and David Lindner. Vision-Language Models are Zero-Shot Reward Models for Reinforcement Learning, October 2023. URL http://arxiv.org/abs/2310.12921. arXiv:2310.12921 [cs].

[17] Kate Baumli, Satinder Baveja, Feryal Behbahani, Harris Chan, Gheorghe Comanici, Sebastian Flennerhag, Maxime Gazeau, Kristian Holsheimer, Dan Horgan, Michael Laskin, Clare Lyle, Hussain Masoom, Kay McKinney, Volodymyr Mnih, Alexander Neitz, Fabio Pardo, Jack Parker-Holder, John Quan, Tim Rocktäschel, Himanshu Sahni, Tom Schaul, Yannick Schroecker, Stephen Spencer, Richie Steigerwald, Luyu Wang, and Lei Zhang. Vision-Language Models as a Source of Rewards, February 2024. URL http://arxiv.org/abs/2312.09187. arXiv:2312.09187 [cs].

[18] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. PaLM: Scaling Language Modeling with Pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023. ISSN 1533-7928. URL http://jmlr.org/papers/v24/22-1144.html.

[19] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and Efficient Foundation Language Models, February 2023. URL http://arxiv.org/abs/2302.13971. arXiv:2302.13971 [cs].

[20] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory

Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, C. J. Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. GPT-4 Technical Report, March 2024. URL `http://arxiv.org/abs/2303.08774`. arXiv:2303.08774 [cs].

[21] Amita Kamath, Jack Hessel, and Kai-Wei Chang. What's "up" with vision-language models? Investigating their struggle with spatial reasoning, October 2023. URL `http://arxiv.org/abs/2310.19785`. arXiv:2310.19785 [cs].

[22] Martin Klissarov, Pierluca D'Oro, Shagun Sodhani, Roberta Raileanu, Pierre-Luc Bacon, Pascal Vincent, Amy Zhang, and Mikael Henaff. Motif: Intrinsic Motivation from Artificial Intelligence Feedback, September 2023. URL `http://arxiv.org/abs/2310.00166`. arXiv:2310.00166 [cs].

[23] Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward Design with Language Models, February 2023. URL `http://arxiv.org/abs/2303.00001`. arXiv:2303.00001 [cs].

[24] Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. MineDojo: Building Open-Ended Embodied Agents with Internet-Scale Knowledge, November 2022. URL `http://arxiv.org/abs/2206.08853`. arXiv:2206.08853 [cs].

[25] Taewook Nam, Juyong Lee, Jesse Zhang, Sung Ju Hwang, Joseph J. Lim, and Karl Pertsch. LiFT: Unsupervised Reinforcement Learning with Foundation Models as Teachers, December 2023. URL http://arxiv.org/abs/2312.08958. arXiv:2312.08958 [cs].

[26] Zhaojing Yang, Miru Jun, Jeremy Tien, Stuart J. Russell, Anca Dragan, and Erdem Bıyık. Trajectory Improvement and Reward Learning from Comparative Language Feedback, October 2024. URL http://arxiv.org/abs/2410.06401. arXiv:2410.06401.

[27] William Chen, Oier Mees, Aviral Kumar, and Sergey Levine. Vision-Language Models Provide Promptable Representations for Reinforcement Learning, May 2024. URL http://arxiv.org/abs/2402.02651. arXiv:2402.02651 [cs].

[28] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep Reinforcement Learning from Human Preferences. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/d5e2c0adad503c91f91df240d0cd4e49-Abstract.html.

[29] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, August 2018. URL http://arxiv.org/abs/1801.01290. arXiv:1801.01290 [cs].

[30] Kimin Lee, Laura Smith, and Pieter Abbeel. PEBBLE: Feedback-Efficient Interactive Reinforcement Learning via Relabeling Experience and Unsupervised Pre-training, June 2021. URL http://arxiv.org/abs/2106.05091. arXiv:2106.05091 [cs].

[31] Arpad E. Elo and Sam Sloan. *The rating of chessplayers : past and present*. Ishi Press International, 2008. URL https://cir.nii.ac.jp/crid/1130282270181653248.

[32] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes, May 2014. URL http://arxiv.org/abs/1312.6114. arXiv:1312.6114 [cs, stat].

[33] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym, June 2016. URL http://arxiv.org/abs/1606.01540. arXiv:1606.01540 [cs].

[34] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Avnish Narayan, Hayden Shively, Adithya Bellathur, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning, June 2021. URL http://arxiv.org/abs/1910.10897. arXiv:1910.10897 [cs].

[35] Wei Zhong and Manasa Bharadwaj. S3D: A Simple and Cost-Effective Self-Speculative Decoding Scheme for Low-Memory GPUs, June 2024. URL http://arxiv.org/abs/2405.20314. arXiv:2405.20314 [cs].

[36] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017. URL http://arxiv.org/abs/1412.6980. arXiv:1412.6980 [cs].

[37] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. February 2017. URL https://openreview.net/forum?id=Sy2fzU9gl.

[38] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning Robust Visual Features without Supervision, February 2024. URL http://arxiv.org/abs/2304.07193. arXiv:2304.07193.

A Implementation details

A.1 Soft-Actor Critic

The implementation of the Soft-Actor Critic (SAC) [29] used in the paper follows the implementation of SAC in PEBBLE [30], similar to Wang et al. [11] to ensure a fair comparison. As such, the used hyperparameters, including the used optimisers, are the same and can be found in the original paper [30].

A.2 Feature extractor

As mentioned in the paper, we train a visual feature extractor $\psi(\cdot)$ using a variational autoencoder (VAE) training objective. In particular, the VAE is implemented using a simple convolutional neural network architecture, the details of which are shown in Table 1. The dimensionality of latent states $|z|$ was chosen to be 16, thus ensuring high informational bottleneck to encourage feature disentanglement. The feature extractor is trained using Adam optimiser [36] with a learning rate of 0.0001. The batch size is 128. A gradient step is applied after each environment step, and the training continues throughout RL training. We set the beta parameter $\beta = 0.1$, which is a common way to encourage better image reconstructions in VAEs [37], and use MSE as reconstruction loss.

To ensure training stability with respect to the RL target (top-rated candidate goal), we fix the weights of the VAE for calculating visual embeddings and update them every $L = 5000$ steps – at the same rate as top-rated candidate goals are updated as RL target states.

Table 1: Architecture of the convolutional feature extractor.

Component	Layer	Output Shape
Encoder	`Conv2D(3, 16, 4x4, stride=2, pad=1) + ReLU`	128x128x16
	`Conv2D(16, 32, 4x4, stride=2, pad=1) + ReLU`	64x64x32
	`Conv2D(32, 64, 4x4, stride=2, pad=1) + ReLU`	32x32x64
	`Conv2D(64, 128, 4x4, stride=2, pad=1) + ReLU`	16x16x128
	`Conv2D(128, 256, 4x4, stride=2, pad=1) + ReLU`	8x8x256
	`Conv2D(256, 256, 4x4, stride=2, pad=1) + ReLU`	4x4x256
	`Flatten`	4096
`Encoder (FC)`	`Linear(4096 → 16)`	16
`Decoder (FC)`	`Linear(16 → 4096)`	4096
Decoder	`ConvT2D(256, 256, 4x4, stride=2, pad=1) + ReLU`	8x8x256
	`ConvT2D(256, 128, 4x4, stride=2, pad=1) + ReLU`	16x16x128
	`ConvT2D(128, 64, 4x4, stride=2, pad=1) + ReLU`	32x32x64
	`ConvT2D(64, 32, 4x4, stride=2, pad=1) + ReLU`	64x64x32
	`ConvT2D(32, 16, 4x4, stride=2, pad=1) + ReLU`	128x128x16
	`ConvT2D(16, 3, 4x4, stride=2, pad=1) + Sigmoid`	256x256x3

A.3 GoalLadder

A.3.1 Candidate goal rating

When a candidate goal gets added to a goal buffer $\mathcal{B}_g$, it gets assigned an initial rating, $\hat{e}$. The initial rating is calculated by taking the mean rating of all existing goals in the buffer, $\hat{e} = \sum_{e_i \in \mathcal{B}_g}^{N} e_i / N$. This ensures that new goals are able to quickly catch up with top-rated goals, while not being immediately considered superior upon initial discovery.

We also set the parameters of ELO rating updates: $C = 400$ (constant controlling how sensitive the expected scores are to rating differences) and $T = 32$ (rating update speed). These were chosen to match the commonly used values in chess rating. In practice, we find that these default parameters were satisfactory for effectively absorbing noisy VLM outputs, while bringing the best goal to the top of the buffer.

A.3.2 Reward definition

Rewards are calculated using Euclidean distance between visual embeddings, $\|\psi(o) - \psi(o^*)\|_2$. In practice, we find that it is useful to normalise rewards within bounds $[0, 1]$, using a simple max-min normalisation, and then apply a non-linear transformation, such that the rewards used during training, $\hat{r}$ are calculated as $\hat{r} = r^\gamma$, where $\gamma = 20$. This ensures that reward scaling does not significantly impact the learning process of the RL agent, as the visual feature extractor changes over the course of training. The non-linear transformation ensures that the agent accrues proportionally larger amounts of reward the closer it gets to the target state.

A.4 Computational resources

For training GoalLadder, we use Tesla V100 16GB GPU and 2× Intel Xeon E5-2698 v4 CPUs, each with 20 cores and 2 hardware threads per core. The training time of a single GoalLadder agent took ∼45 hours. Initial experimentation included identifying the abilities of the used vision-language model, Gemini 2.0 Flash, to compare pairs of visual observations with respect to a language instruction.

B Task descriptions and prompts

Below is the prompt template used to get feedback from a vision-language model. The formatting instructions of the expected response can be adjusted depending on the implementation.

Further, Table 2 shows the used language instructions for each task in GoalLadder. The same template is used for RL-VLM-F for controlled comparison. Table 3 shows language instructions used for reward calculation in VLM-RM. Finally, Table 4 shows language instructions used in RoboCLIP.

VLM prompt template in GoalLadder

Image 1: `{IMAGE 1}`
Image 2: `{IMAGE 2}`

The goal `{LANGUAGE INSTRUCTION}`.

Answer the following questions:
1. What is shown in Image 1?
2. What is shown in Image 2?
3. Is there any difference between Image 1 and Image 2 in terms of achieving the goal?
4. Is the goal better achieved in Image 1 or in Image 2? Explain your reasoning.

If the goal is better achieved in Image 2 than it is in Image 1,
`{FORMATTING INSTRUCTIONS}`.

Table 2: GoalLadder language instructions for each task

Task	LANGUAGE INSTRUCTION
cartpole	"is pole balanced upright"
mountaincar	"is car at the peak of the mountain, to the right of the yellow flag"
drawer-open-v2	"of the robotic arm is to open the green drawer"
drawer-close-v2	"of the robotic arm is to close the green drawer"
sweep-into-v2	"of the robotic arm is to sweep the green cube into the hole"
window-open-v2	"of the robotic arm is to open the window"
button-press-topdown-v2	"of the robotic arm is to press the red button into the orange box"

Table 3: VLM-RM language instructions for each task

Task	LANGUAGE INSTRUCTION
cartpole	"pole vertically upright on top of the cart"
mountaincar	"car at the peak of the mountain, to the right of the yellow flag"
drawer-open-v2	"open drawer"
drawer-close-v2	"closed drawer"
sweep-into-v2	"green cube in a hole"
window-open-v2	"open window"
button-press-topdown-v2	"pressed button"

Table 4: RoboCLIP language instructions for each task

Task	LANGUAGE INSTRUCTION
drawer-open-v2	"robot opening drawer"
drawer-close-v2	"robot closing drawer"
sweep-into-v2	"robot pushing green cube into a hole"
window-open-v2	"robot opening window"
button-press-topdown-v2	"robot pressing red button"

C Additional ablations

C.1 ELO rating

To disentangle the contributions of the ELO ranking more thoroughly, we have run a small ablation study of GoalLadder's performance without the ELO system, i.e. the 'greedy' rating system where we always trust the VLM's decision to replace the top-rated goal. Table 5 shows the results on the *Drawer Open* task, confirming the importance of the ELO rating system in achieving good performance.

Table 5: ELO ablation.

GoalLadder	**Final success rate**
with ELO	0.97 ± 0.11
without ELO	0.20 ± 0.35

Qualitatively, we observe the expected suboptimal behaviour in GoalLadder without ELO: significantly better goals are periodically replaced by significantly worse goals as targets (i.e. VLM makes mistakes). This destabilises the training process and results in a rapid performance drop.

C.2 Buffer size

In our initial experimentation, we have found that the buffer size should be big enough to allow for diversity, but small enough to allow for frequent pairwise comparisons between the candidate goals.

Table 6: Success rates by buffer size.

Buffer size	Final success rate	Average success rate	Max success rate
1 (No ELO)	0.20 ± 0.35	0.14 ± 0.23	0.25 ± 0.45
5	0.83 ± 0.13	0.48 ± 0.13	1.00 ± 0.00
25	0.80 ± 0.11	0.47 ± 0.14	1.00 ± 0.00
50	1.00 ± 0.00	0.61 ± 0.11	1.00 ± 0.00
10000	0.00 ± 0.00	0.01 ± 0.01	0.13 ± 0.27

To confirm this intuition, we have run a small ablation of GoalLadder with varying buffer sizes. The results shown in Table 6 confirmed our initial hypothesis: (a) smaller buffer sizes hinder performance as the diversity of candidate goals in the buffer suffers; (b) large buffer sizes require an exponentially larger number of pairwise comparisons to arrive at converged ratings, thus also hindering the performance of the model. Overall, we find that GoalLadder is robust to buffer sizes within the reasonable ranges.

C.3 Visual encoders

We have similarly tested the performance of GoalLadder with two off-the-shelf visual encoders – DINOv2 [38] and CLIP [5] – shown in Table 7. These results indicate that a small VAE trained on environment observations outperforms large, pre-trained encoders. We hypothesise that the VAE allows the model to better distinguish between fine-grained state differences, which are not well captured by the pre-trained models.

Table 7: Success rate by vision encoder.

Model	Max success rate
GoalLadder + VAE	1.00 ± 0.00
GoalLadder + DINOv2	0.25 ± 0.21
GoalLadder + CLIP	0.38 ± 0.17

D NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We demonstrate that GoalLadder is able to solve continuous control tasks, surpassing the performance of competitor methods and almost matching the oracle. By running competitor methods under the same feedback rate, we also showcase GoalLadder's sample efficiency with respect to the required amount of feedback. Lastly, we present ample evidence of the fact that GoalLadder's rating system consistently brings the best goal in the environment to the top of the ranking, thus reducing the effects of noisy VLM feedback.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the fact that GoalLadder is currently only applicable to tasks with static goals. Furthermore, we discuss the potential limitation of using visual similarity measure for calculating rewards.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: [NA]

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Full implementation details are included, both in the main body and the appendix of the paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code will be provided.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Specified in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Discussion.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.