Abstract

Building generalist agents that can accomplish many goals in rich open-ended environments is one of the research frontiers for reinforcement learning. A key limiting factor for building generalist agents with RL has been the need for a large number of reward functions for achieving different goals. We investigate the feasibility of using off-the-shelf vision-language models, or VLMs, as sources of rewards for reinforcement learning agents. We show how rewards for visual achievement of a variety of language goals can be derived from the CLIP family of models, and used to train RL agents that can achieve a variety of language goals. We showcase this approach in two distinct visual domains and present a scaling trend showing how larger VLMs lead to more accurate rewards for visual goal achievement, which in turn produces more capable RL agents.

1 Introduction

Many of the biggest successes of reinforcement learning (RL, [25]) have been in domains where a clear reward function was readily available. Reward functions that have been successfully used include game win/loss [21, 28, 30], change in game score [2, 18], change in underlying state like forward motion [15] and negative distance to a desired state configuration. With these successful applications of RL to challenging tasks there has been growing interest in building generalist agents capable of achieving many challenging goals in rich environments.

One of the main limiting factors in applying RL to building generalist agents is the need for many reward functions for different goals. Building a reward function which, when maximized, will lead to the achievement of a particular goal can be challenging, time consuming, and hard to scale [14, 19]. This is true both in simulated environments, where determining whether an agent successfully achieved an abstract goal like building a house from the underlying state variables is difficult to...
express in code, and in the real world, where a reward has to be computed from observations. These challenges have put the spotlight on automatic ways of generating reward functions for training generalist agents [14] [34].

One particularly promising direction that has emerged recently is the use of vision-language models (VLMs) for building reward functions in visual environments [4, 7, 9, 17]. VLMs trained on large datasets of paired images and text have been shown to perform well on a variety of visual detection, classification and question answering tasks out of the box and can act as good initializations for finetuning on specialized tasks or datasets. Recent work showed that a pretrained CLIP model [20] finetuned on paired Minecraft videos and text can be used as an effective shaping reward for training agents to achieve hunting and foraging tasks in Minecraft [9]. Similarly, [7] showed how a pretrained Flamingo [1] VLM can be finetuned on visual question-answering data to produce an accurate visual success detector for a variety of natural language goals.

Encouraged by the success of these approaches we explore the feasibility of using off-the-shelf VLMs to derive accurate reward functions for language goals in visual environments. We propose a way of deriving a sparse binary reward for visual achievement of language goals from pretrained CLIP image and language embeddings and show how it can be used to train agents to achieve a variety of language goals in the Playhouse [27] and AndroidEnv [29] visual environments. We believe that our work presents a compelling demonstration of how off-the-shelf VLMs can be used to train grounded language agents without the need for finetuning on environment specific data.

2 Related Work

2.1 VLMs Rewards

A number of prior works have investigated using CLIP as a reward function for RL. Most closely related to our work is MineDojo [9] which first finetunes CLIP with Minecraft videos to form MineCLIP. Then MineCLIP is used as a dense shaping reward function in addition to a ground truth binary reward function for programmatic tasks where ground truth is available, except for creative tasks where no ground truth is available. The main differences between this work and MineDojo are that (i) we use an off-the-shelf CLIP model, (ii) we do not utilize any ground truth information during RL training, and (iii) we train a single RL agent that can solve thousands of language-based goals. Similar to MineDojo, CLIP4MC [6] finetunes CLIP on a curated dataset of Minecraft Youtube videos. CLIP4MC then uses PPO as the RL training backbone and the policy is trained on the CLIP4MC reward in the same form as MineCLIP reward together with the sparse completion where available. CLIP has also been used for reward-shaping simulated robotics domain [4, 17, 23]. RoboCLIP [23] computes a sparse trajectory level reward as the cosine similarity score between the embedding of the trajectory video and the embedding of the text description of the task.

CLIP encoder was used for semantic intrinsic reward signal for curiosity-driven exploration in [26]. An orthogonal approach for scaling up reward modeling is to use Large Language Models (LLMs) to translate task specification into reward functions executable programs, resulting in sparse rewards [35] or dense rewards [32]. Earlier works also use LLMs itself as binary reward in negotiation games [13], collaborative human-AI interaction games [12], or dense reward based on the cosine similarity of a caption (hard-coded or learned) of transition and goals in a 2D Crafter environment [8, 10].

2.2 VLMs for Hindsight Relabeling

Several works used VLMs for hindsight relabeling to do behaviour cloning. [24] uses Flamingo for hindsight relabeling and then behaviour cloning in the Playhouse environment (similar to our Lift task). DIAL [31] uses CLIP finetuned on robotic demonstration and autonomous data that has been hindsight labeled by human annotators to further relabel a larger set of offline robotic trajectories, then also perform behaviour cloning a language-conditioned policy.

2.3 VLMs as Success Detectors

SuccessVQA [7] finetunes smaller 3B Flamingo models to act as success detectors by formulating the problem as a Visual Question Answering (VQA). Most of their experiments focus on how accurate
Go near a helicopter
Lift the pink boat
Image Encoder
Language Encoder Softmax & threshold
binary reward
Cosine similarity
Goal
Observation (single or multi-frame)
Negatives
Scores

Figure 1: Architecture for Vision-Language Models (VLMs) as rewards. The VLM trained contrastively contains an image encoder $f_\theta$ and language encoder $g_\theta$. We embed the current environment observation frame(s) using the image encoder, along with the desired goal language descriptions $l$ and negative language descriptions using the language encoder. The reward is computed by taking the cosine similarity scores and applying softmax and thresholding.

the success detector is on offline datasets across various domains (robotics, Playhouse, Ego4D), with one experiment using the SuccessVQA model for reward-filtered behaviour-cloning.

3 Method

In this work we investigate how to leverage contrastive VLMs like CLIP to produce text-based reward models. Our aim is to produce a reward function that, given a language-based goal and an image observation, produces a scalar reward corresponding to whether or not that goal has been achieved. Concretely, we construct a reward model which consists of an image encoder $f_\theta(o)$ and text encoder $g_\theta(l)$ where $o \in O$ is an observation and $l \in \mathcal{L}$ is a text-based goal. The reward model inputs both observation and goal $r(o, l)$ and outputs a binary reward: 1 if the goal has been achieved and 0 otherwise.

We operate in a partial observed Markov Decision Process (POMDP) setting where the agent sees observations $o_t \in O$, acts in an environment with actions $a_t \in A$, observes the next observation according to the environment dynamics $o_{t+1} \sim \mathcal{P}(o_{t+1}|a_t, o_t)$, and receives a reward $r_t \sim \mathcal{R}$ with a discount factor $\gamma \in (0, 1]$. In addition to the standard POMDP, our agent is given a language-based goal $l \sim \mathcal{L}$ for the duration of the episode and the reward is computed via a VLM as a function $r_t = r(o_{t+1}, l)$. The episode terminates either on timeout $T$ or when $r_t = 1$ meaning that the goal has been achieved according to the intrinsic VLM reward function. Note that we assume no access to the ground truth reward during training, and only use the ground truth reward for evaluation.

3.1 Turning CLIP into a reward function

We first compute the probability $p_{o_{t+1}}$ that the agent achieves the goal given by language description $l$ after acting in the environment, out of a set of potential goals $l' \in \mathcal{L}$ in the task set $\mathcal{L}$, by applying softmax with temperature $\tau$ over the cosine similarity between the embedding of the state, $f_\theta(o_t)$, and the embedding of the language description $g_\theta(l)$ across the set of potential goals:

$$p_{o_{t+1}} = \frac{\exp(f_\theta(o_t) \cdot g_\theta(l)/\tau)}{\sum_{l'} \exp(f_\theta(o_t) \cdot g_\theta(l')/\tau)}$$

We then convert this reward function into a binary reward function by thresholding the probability on the hyperparameter value $\beta \in [0, 1]$:

$$r_t = r(o_{t+1}, l) = \mathbb{I}[p_{o_{t+1}} > \beta]$$

where $\mathbb{I}$ is an indicator function. To avoid stochastic rewards, we sample negatives uniformly from the task set $l \in \mathcal{L}$ and fix them throughout the duration of RL training. In our experiments, the task
set is a discrete set of language goals that are known to be achievable in the environment. We leave investigations on generalizing the negatives sampling from generative distribution, such as samples from an LLM, for future work.

4 Experiments

In our experiments, we aim to answer the following questions:

1. Does maximizing the VLM reward maximize the ground truth reward? (Section 4.4.1)
2. How does scaling the size of the VLM affect the performance of the VLM reward? (Section 4.4.2)
3. Can online agent performance be predicted from offline evaluation metrics? (Section 4.4.2)
4. What are the effects of prompt engineering on the performance of the VLM reward? (Section 4.4.3)

4.1 Experimental Setup

Our experimental setup is similar to that of standard online RL where an agent maximizes returns via trial-and-error interactions with an environment. The only difference is that rather than training against a ground truth or hand-engineered reward function, our agent trains against rewards emitted by a VLM. For evaluation purposes, we plot both the VLM rewards, which we call intrinsic, and ground truth rewards, which are provided by access to the simulator. We train a language-conditioned policy that is initialized randomly, while the reward model is a pre-trained CLIP VLM whose weights are frozen. We use Muesli [11] as the RL algorithm to optimize the policy against the VLM reward function. The environment samples a goal from the set of training goals, and the agent is tested on held-out goals during the evaluation.

4.2 CLIP architecture

We use pre-trained CLIP [20] models for all experiments. We investigate two types of encoder architectures for CLIP. The first utilizes Normalizer-Free Networks (NFNets) [3], which uses a ResNet image encoder with adaptive gradient clipping in lieu of batch normalization. The second is Swin [16, 33], which is a hierarchical vision transformer architecture for image encoding. We use the BERT architecture [5] as the language encoder in all CLIP models.

4.3 Environments and tasks

We experimented with two domains for visual environments: (1) Playhouse [27], and (2) AndroidEnv [29]. Figure 2 illustrates example observations from the two domains and their tasks. Playhouse is a Unity-based environment with an egocentric view inside a procedurally-generated home. We experimented with three families of tasks of increasing difficulty: (1) Find, (2) Lift, and (3) Pick and Place. In the Find task, the goal is to locate the object and approach the object as closely as possible. In the Lift task, the agent must also grab the object and lift it. Finally, in the Pick and Place task, the agent needs to pick up the desired object and place it near another specified object.

Figure 2: Environments and example tasks. (Left) Playhouse [27] consists of Find, Lift, and Pick and Place tasks. (Right) AndroidEnv [29] consists of opening app tasks across various apps on Android.
Figure 3: Performance of an agent over the course of online reinforcement learning training when optimizing against the learned VLM reward. We measure both the (1) learned VLM reward return during training and (2) ground truth reward on held-out evaluation tasks. There is strong correlation between optimizing the learned VLM reward and the ground truth reward.

AndroidEnv \cite{29} is an open-source environment built on the Android operating system, allowing the agent to interact through touchscreen gestures on an emulated Android device in real-time. is an Android device emulator that can be computer controlled. The observation space consists of realtime RGB images of the Android screen. The action space consists of swipe, touch, and type actions and mimics the way a human would interact with the device. We run experiments within two task sets within this environment. The first task, Open Common Apps, involves opening one of ten common apps such as Gmail, Google Sheets, Chrome, Calculator, Clock, Messages, Phone, Google Calendar, and Google Docs. The second task, Open Diverse Apps, expands the common app task set with fifty additional less well-known apps.

4.4 Experimental Results

4.4.1 Maximizing learned rewards maximizes ground truth

For each of the environments and task families, we train a language goal-conditioned agent to maximize purely the VLM reward, without observing the ground truth reward. During the course of training, we also log the ground truth reward that the agent would have received. We observe in Figure 3 that the agent trained to maximize reward also maximizes the ground truth reward. Since the VLM reward is not perfectly accurate, there is a systematic gap between the VLM reward versus the ground truth reward. However, within the training budget, we did not observe reward hacking \cite{22} of our VLM reward, where the true reward drops off while the proxy VLM reward continues to increase.

Figure 4: Scaling reward model size. (Left) Precision-Recall curves for varying VLM architecture and sizes on an offline fixed dataset of Playhouse trajectories. (Right) Ground truth returns on held-out evaluation tasks for Playhouse over the course of training with varying VLM reward sizes.
4.4.2 Scaling reward function model

We investigate the scaling behaviour of the reward model as a function of the size of the underlying VLM. Firstly, we evaluate the accuracy of the VLM reward against ground truth binary reward on an offline, fixed dataset of Playhouse trajectories. Figure 4 (left) shows precision-recall curves obtained by a selection of models (CLIP with NFNet [3] and Swin [16, 33] vision encoders) evaluated against our Playhouse task set. The sensitivity is adjusted by varying the threshold $\beta$ above which the reward is given. We observe that increasing the size of the VLM used for the reward model (from 200M to 1.4B parameters) improves the precision-recall curves. Figure 4 (right) shows the ground truth returns for held-out evaluation tasks over the course of training which are not given to the agent, when trained with VLM reward signals with different base models. We observe that the improved accuracy of the VLMs on offline datasets, when used as the only reward signal, does translate to better agent performance on ground truth evaluation metrics.

4.4.3 Prompt Engineering VLM Reward

Manually prompt-engineering the text template has been found to improve the zero-shot classification of CLIP [20]. Similarly, we found that prompt-engineering the text template for our VLM reward can have a significant impact on the ground truth performance of the agent policies trained on the VLM reward. Figure 5 (left) compares the ground truth return evaluation of agent policies trained on various prompt templates (Figure 5 (right)). We hypothesize that, due to the training distribution of CLIP image captions, using action verbs such as “open” in the prompt is not as helpful as providing a description of the successful states, such as “Screenshot of [TASK] on Android”.

5 Conclusion

This work shows how accurate rewards for visual achievement of language goals can be derived from off-the-shelf VLMs like CLIP without finetuning on domain-specific datasets. Our scaling trend analysis showed that the quality of the rewards derived from CLIP-style models increases as the number of parameters in the visual encoder grows. Taken together, we believe these results suggest that as the quality of VLMs improves it may be possible to train generalist RL agents in rich visual environments without the need for any task or domain-specific finetuning of the reward model.

6 Authors and Contributions

We list authors alphabetically by last name. Please direct all correspondence to Volodymyr Mnih (vmnih@google.com), Feryal Behbahani (feryal@google.com), Michael Laskin (mlaskin@google.com) and Harris Chan (harrischan@google.com).
6.1 Full-time Contributors:

- **Kate Baumli**: agent research, infrastructure engineering
- **Satinder Baveja**: advising
- **Feryal Behbahani**: research vision, team leadership, agent research
- **Harris Chan**: reward function research, paper writing
- **Gheorghe Comanici**: agent research
- **Sebastian Flennerhag**: agent research
- **Maxime Gazeau**: reward function research
- **Dan Horgan**: engineering leadership
- **Michael Laskin**: reward function research, paper writing
- **Volodymyr Mnih**: research vision, team leadership, reward function research, paper writing
- **Alexander Neitz**: agent research, evaluation
- **Fabio Pardo**: reward function research
- **John Quan**: agent research
- **Himanshu Sahni**: reward function research, evaluation
- **Tom Schaul**: agent research
- **Yannick Schroecker**: agent research, evaluation
- **Stephen Spencer**: infrastructure engineering, evaluation
- **Richie Steigerwald**: evaluation, infrastructure engineering
- **Luyu Wang**: reward function research, infrastructure engineering
- **Lei Zhang**: agent research

6.2 Part-time Contributors:

- **Kristian Holsheimer**: infrastructure engineering
- **Clare Lyle**: agent research
- **Kay McKinney**: project management
- **Hussain Masoom**: project management
- **Jack Parker-Holder**: agent research
- **Tim Rocktäschel**: advising

References


