IRIS AperTO

UNIVERSITÀ DEGLI STUDI DI TORINO

AperTO - Archivio Istituzionale Open Access dell'Università di Torino

Vision Language Models as Policy Learners in Reinforcement Learning Environments

This is a pre print version of the following article:

Original Citation:

Availability:

This version is available http://hdl.handle.net/2318/2030690 since 2024-11-15T11:13:42Z

Publisher:

i6doc

Published version:

DOI:10.14428/esann/2024.es2024-181

Terms of use:

Open Access

Anyone can freely access the full text of works made available as "Open Access". Works made available under a Creative Commons license can be used according to the terms and conditions of said license. Use of all other works requires consent of the right holder (author or publisher) if not exempted from copyright protection by the applicable law.

(Article begins on next page)

16 October 2026

Vision Language Models as Policy Learners in Reinforcement Learning Environments

Giovanni Bonetta[1], Davide Zago[2], Rossella Cancelliere[2]
Mirko Polato[2], Bernardo Magnini[1] *

1- Bruno Kessler Foundation, via Sommarive, 18 - Povo 38123 Trento, Italy
2- University of Turin - Department of Computer Science
via Pessinetto 12 - 10149 Turin, Italy

Abstract.
In various domains requiring general knowledge and agent reasoning, traditional reinforcement learning (RL) algorithms often start from scratch, lacking prior knowledge of the environment. This approach leads to significant inefficiencies as agents must undergo extensive exploration before optimizing their actions. Conversely, in this paper we assume that recent Vision Language Models (VLMs), integrating both visual and textual information, possess inherent knowledge and basic reasoning capabilities, offering potential solutions to the sample inefficiency problem in RL. The paper explores the integration of VLMs into RL by employing a robust VLM model, Idefics-9B, as a policy updated via Proximal Policy Optimization (PPO). Experimental results on simulated environments demonstrate that utilizing VLMs in RL significantly accelerates PPO convergence and improves rewards compared to traditional solutions. Additionally, we propose a streamlined modification to the model architecture for memory efficiency and lighter training, and we release a number of upgraded environments featuring both visual observations and textual descriptions, which, we hope, will facilitate research in VLM and RL applications.

1 Introduction

Reinforcement learning (RL) agents' policies are usually learned from scratch and updated according to the rewards from the environment [1], which ensures that RL agents are well aligned with it. However in situations requiring agent's reasoning and planning abilities, algorithms starting from a *tabula rasa* state initiate a exploration-exploitation loop: the agent must first learn the rules of the environment, which situations are favorable or not, which states are more probable or less (exploration), and then use the acquired knowledge to maximize its own advantage (exploitation). The less environment steps are needed in order for an agent to learn, the more sample efficient the algorithm is.

A recent successful approach to get better sample efficiency exploits Large Language Models (LLMs) in RL. LLMs demonstrated significant success in natural language generation and understanding [2, 3], also thanks to the knowledge they make accessible through language. Recent studies show that the huge

*We acknowledge ISCRA for awarding this project access to the LEONARDO supercomputer, owned by the EuroHPC Joint Undertaking, hosted by CINECA (Italy).

amount of data used for their training gives them the ability to play sophisticated games such as TextWorld [4], Handbi [5], and MineCraft [6], allowing the emergence of basic reasoning skills and the creation of simple action plans [7]. Also the well known misalignment issues [8], which sometimes cause LLMs' failure in solving simple decision-making tasks, can be effectively addressed by leveraging RL to align LLMs with embodied environments [9, 10].

While LLMs are being successfully integrated in RL scenarios, their input remain textual, which requires to convert visual content from the environment into complex language descriptions, sometimes too elaborated to be effectively exploited. A step forward would be to take advantage of a visual model to jointly use the textual description and the visual content of the environment. Accordingly, the contribution of this paper is threefold: firstly we use Idefics-9B [11], a recent Visual Language Model (VLM) whose component of visual knowledge is exploited for solving the FrozenLake gymnasium environment and some typical procedurally-generated minigrid games. Besides, we propose a straightforward method to adapt the model's architecture in order to save memory and streamline training, and we release some enhanced environments that incorporate not only visual data but also their textual descriptions in English.

2 Our Method

We assume a Reinforcement Learning scenario [1] formulated as a Markov Decision Process (MDP) [12] defined by the tuple (S, A, T, R, γ), where S is the state space, A the action space, T the transition dynamics, R the reward function and γ the discount scalar. In this context our first contribution is to create an agent, which acts following a policy $\hat{\pi}$ generated by the underlying VLM, that needs to maximize the expected cumulative reward.

We train and test our agent in embodied gridworld-like environments, where it needs to plan and act to achieve a specific goal provided by the game. The environment simulators follow the Gymnasium API [13] and provide at each timestep t a textual description y_t of the current state (e.g., what the agent sees, if it is carrying an object, what the goal is, etc...) and a RGB visual observation x_t of the same view. Such information is organized in prompts and fed to the model so that it can choose how to proceed, selecting one among the environment's action set $A = \{\alpha_1, ..., \alpha_i, ..., \alpha_n\}$. Each action α_i has a textual description a_i, where a_i is the sequence of tokens $\{w^i_1, .., w^i_{|a_i|}\}$.

The training loop is performed using the PPO algorithm [14], which aims to concurrently optimize a policy $\hat{\pi} : S \to \mathbb{P}(A)$ and a value function $\hat{V} : S \to \mathbb{R}$. We introduce a value head on top of the last VLM's hidden state and use the (log)-likelihood of the action tokens as policy.

Using a VLM as a policy network in PPO requires the model to generate a probability distribution over the possible actions in the environment. We formulate this problem as a multiclass classification task in which the model, for every time step t (hereon omitted in suffix for brevity), has to choose among different prompts, each tailored to a specific action. Each prompt $U^i = \{u_1, ..., u_{|U^i|}\}$ is

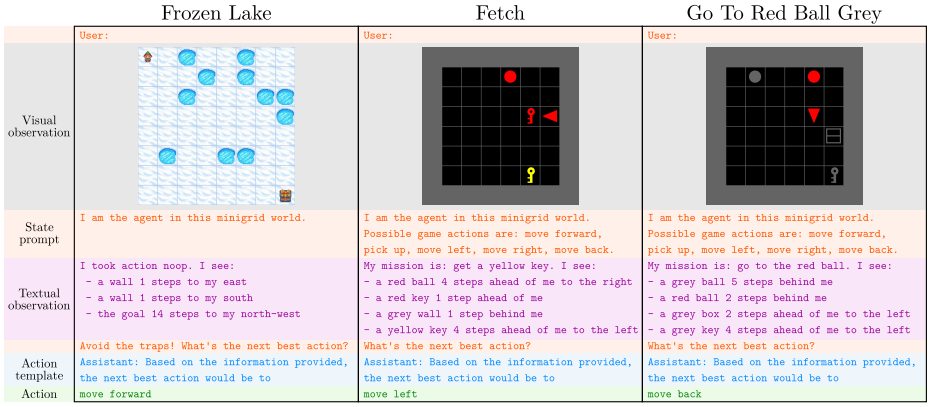

Fig. 1: Prompt example. The prompt is made of the State prompt concatenated with the Action prompt. The State prompt contains the Textual descriptor and the Visual observation, while the Action prompt contains the action description.

made by concatenating two templates:

- the state prompt $P = \{p_1, ..., p_k\}$. A general fixed text template populated by the game state descriptors x_t and y_t.[1]
- the action prompt $Q = \{q_1, ..., q_z\}$. A general fixed text template populated by one specific action description a_i.

The VLM is fed with each prompt and outputs the log-likelihood of the prompt as the average of the token logits that compose it. The log-likelihood of U^i is:

$$\log \mathbb{P}(U^i|P,Q) = \frac{1}{|U^i|} \sum_{j=1}^{|U^i|} \log \mathbb{P}(u^i_j|P,Q,u^i_{<j}) \tag{1}$$

The subsequent softmax normalization of $\log \mathbb{P}(U^i|P,Q)$ provides a valid probability distribution over the action set A.

To compute the state value, we add a value head, i.e., a simple multilayer perceptron with one output, on top of the last token in the state prompt. The primary advantage of using (1) is that the agent can exploit the language and visual backbones' priors of the VLM and its implicit knowledge about the world. For example, if the agent in a given game state has a death cell on his right, we argue that, due to the huge amount of pre-training data, the VLM has learnt to avoid bad states and consequently will output an higher logit for going left, instead of falling to death. A downside of (1) is that it requires to feed the model with a mini-batch composed of as many prompts as the number of actions. This can quickly lead to memory issues as the number of actions or the prompt lengths increase. As a countermeasure we use LoRA [15] paired with

[1] x_t is not a sequence of tokens, but rather a 3D tensor. Nonetheless it is transformed into a sequence of feature tokens within the model's forward pass and interleaved with the text content.

model quantization techniques: so doing we reach the goal of saving memory and perform faster trainings.

We load the model with `int4` quantization using BitsAndBytes [2], and freeze its parameters during training. The 80 millions trainable weights are the LoRA injected layers, which counts for the 0.87% of the total plus the negligible value head. This setup allows us to parallelize our trainings on four A40 (65GB) GPUs, fitting one syncronized instance of the model in each of them. The parallelization and the synchronization of the gradients among the instances is managed by the Accelerate library [3]. Our code will be made available after publication.

3 Experiments and Results

3.1 Environments

The agents are trained in two typical procedurally-generated grid environments:

Frozenlake. [13] The agent is spawned in the upper-left corner of a 8x8 grid world and needs to reach the goal located to the bottom-right corner. In order to win the episode, the agents must avoid falling into the icy ponds randomly scattered on his way. The episodic reward is 1 if the agent reaches the goal safely, otherwise the rewards is always 0. This environment comes with 4 possible actions: move-{west/south/east/north}.

Minigrid & BabyAI. [16, 17] The agent is spawned in a random location and needs to complete a specific task described in plain text. In *MiniGrid-Fetch* the agent is surrounded by items of different types and colors and must pick up a specific item. In *BabyAI-GoToRedBallGrey* the agent has to pick up the red ball and the grid is populated with useless items which acts as visual and textual distractors. Both games have the following possible actions: move-{forward/left/right/back} and pick up. The reward function is also the same: 0 for picking up the wrong item, otherwise $1 - 0.9(T_\tau/T_{max})$, where T_τ and T_{max} are the number of steps of the episode and the maximum number of steps, respectively.

All the considered environments have sparse rewards. This is known to be a challenge since the agent gets a positive feedback only when it successfully achieves the goal. Since these simulators come with no textual description of the game state, as a paper contribution we design rule-based scripts to convert the symbolic representation of a scene into a english textual description, which we hope to be useful for research purposes in the field of RL with VLM systems.

3.2 Results

We compare our system called *Idefics-Agent* with: *Idefics* (the non PPO-finetuned version), a commonly used CNN baseline adapted from CleanRL [18] that we call *CNN-Agent*, and the *Random-Agent* which acts randomly in the environments.

[2] `https://github.com/TimDettmers/bitsandbytes`
[3] `https://huggingface.co/docs/accelerate/index`

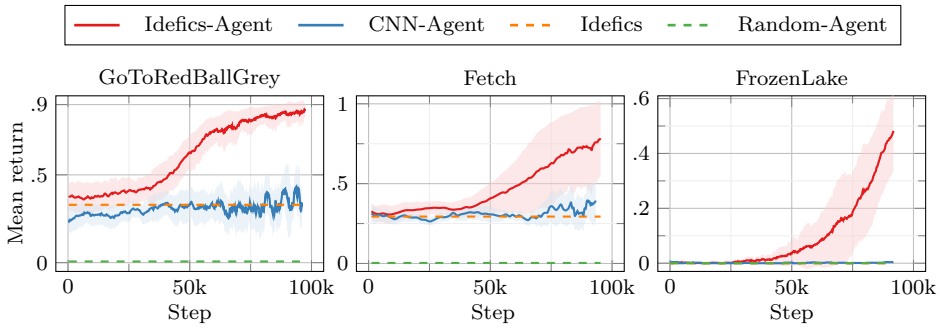

Fig. 2: Mean reward performance on the three environments.

Experiments are repeated with three different seeds for statistical significance. Figure 2 shows the average models' performance with standard deviations bands.

Our experiments are designed to answer three questions:

1) Is a VLM able to achieve better performance than a solution starting from a tabula-rasa state? The plots show that Idefics-Agent surpasses CNN-agent and Random-Agent in terms of mean reward and needs less experience to learn: 100k steps are enough to show convergence on a good policy for all the environments. The CNN-Agent, on the other hand, does not converge in any of them with the same amount of experience. As previously mentioned, our environments, although requiring simple skills to be solved like reaching an object or avoiding death states, are challenging for their sparse reward settings. Tabula-rasa methods, as the CNN we used, may have to explore massively before reaching a positive reward state, and are therefore penalized.

2) Is the VLM's implicit knowledge important for performance? Comparing Idefics with the Random-agent we see that, even without training, Idefics acts better than a random policy, reaching higher expected returns. We experimented with different state and action template pairs and saw that (*"I am the agent in this minigrid world.* {} *What's the next best action?"*, *"Based on the information provided, the next best action would be to* {}*"*)[4] is already enough to elicit good action biases.

3) Is the PPO training necessary to align the VLM on decision-making environments? Comparing Idefics-Agent with Idefics (i.e. without PPO finetuning) we see that the latter does not have all the skills required to solve the environments, which proves PPO finetuning to be necessary to have strong performance. Note that Idefics is comparable to the CNN-Agent with 100k steps of training, at least in GoToRedBallGrey and Fetch environments.

[4]The {} within the templates indicate where the observations and actions, for value and action template respectively, are injected.

4 Conclusions

In this work we showed how to use a strong VLM, Idefics-9B, as a learning policy in interactive reinforcement learning environments. We tested our Idefics-Agent on different environments that we augmented to provide textual observations besides visual ones. Furthermore, we modified the architecture of the model for a light and efficient PPO training utilizing LoRA and quantizations. Finally we showed strong results in terms of rewards and sample efficiency, supporting the hypothesis that it is indeed possible to build on VLM's latent knowledge to take decision-making scenarios. Delving deeper in aspects related to VLM's vs LLM's specific knowledge will be addressed as future work.

Acknowledgements
We acknowledge the support of the PNRR project FAIR - Future AI Research (PE00000013), under the NRRP MUR program funded by the NextGenerationEU.

References

[1] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.

[2] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[3] OpenAI. Gpt-4 technical report, 2023.

[4] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, et al. React: Synergizing reasoning and acting in language models. *arXiv:2210.03629*, 2022.

[5] Hengyuan Hu and Dorsa Sadigh. Language instructed reinforcement learning for human-AI coordination. *arXiv preprint arXiv:2304.07297*, 2023.

[6] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, et al. Voyager: An open-ended embodied agent with large language models. *arXiv:2305.16291*, 2023.

[7] Yue Wu, So Yeon Min, Shrimai Prabhumoye, et al. Spring: Studying papers and reasoning to play games. In *NeurIPS*, volume 36, pages 22383–22687. Curran Associates, Inc., 2023.

[8] Anthony Brohan et al. Do as I can, not as I say: Grounding language in robotic affordances. In *Proc. 6th Conference on Robot Learning*, pages 287–318, 2023.

[9] Thomas Carta et al. Grounding large language models in interactive environments with online reinforcement learning. In *ICML*, 2023.

[10] Weihao Tan, Wentao Zhang, et al. True knowledge comes from practice: Aligning llms with embodied environments via reinforcement learning. *arXiv:2401.14151*, 2024.

[11] Hugo Laurençon, Lucile Saulnier, Léo Tronchon, Stas Bekman, et al. Obelics: An open web-scale filtered dataset of interleaved image-text documents. *arXiv:2306.16527*, 2023.

[12] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

[13] Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, et al. Gymnasium, March 2023.

[14] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.

[15] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, et al. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

[16] Maxime Chevalier-Boisvert et al. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *CoRR*, abs/2306.13831, 2023.

[17] Maxime Chevalier-Boisvert et al. Babyai: A platform to study the sample efficiency of grounded language learning. *arXiv:1810.08272*, 2018.

[18] Shengyi Huang and other. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *JMLR*, 23(274):1–18, 2022.