Proceedings of Machine Learning Research 260, 2024 ACML 2024

Supplementary Materials of 'Exploring Beyond Curiosity Rewards: Language-Driven Exploration in RL'

Editors: Vu Nguyen and Hsuan-Tien Lin

Appendix A. Limitations and Discussions

A limitation of the present study that is worth acknowledging lies in the computational cost associated with LLMs. While our approach significantly enhances sample efficiency, it is important to recognize that querying massive LLMs may introduce an additional overhead in terms of time and computation. To mitigate this, a strategy could involve distilling the LLM into more compact actor networks, similar to Parisotto and Salakhutdinov (2021). In addition, strategies such as quantization, limiting the number of output tokens, or staged speculative decoding Spector and Re (2023) can be employed to accelerate their inference. It is important to note, however, that these computational constraints do not significantly impact the action generation task due to the minimal number of tokens it generates.

As pointed out in some recent papers Nair et al. (2018), exploring the task following sub-optimal demonstrations often results in poor performance or a suboptimal policy Du et al. (2023). Conversely, distilling a generative model through an exploration bonus has been shown to enable learning better-than-expert performance, even from a noisy expert Yu et al. (2020). Thus, iLLM relies on curiosity-driven exploration to enable better-than-llm performance. Morevoer, this design also offers modularity, allowing the integration of any language tasks such as summarization or planning to guide the agent's exploration.

In the action generation task, we select actions based on the highest log probability rather than generating actions directly. A significant challenge with direct action generation is the variability in the format of the generated actions, necessitating an additional step to map these actions to the environment's action space. By selecting the action with the highest log probability, we bypass this step. Besides, the selected environments did not require complex action descriptions, alleviating the need to generate actions represented by sentences. We leave to future work to explore this direction further.

Another avenue for improvement is to leverage self-reflection on text observations and *translated* state-action pairs. Self-reflection mechanisms Park et al. (2023) that enable the agent to assess the quality and informativeness of textual observations could refine the summarization process. By employing self-assessment, the agent may learn to prioritize and extract key information, ultimately leading to more concise and informative textual representations.

We also aim to explore alternative formulations of intrinsic rewards. In our current implementation, these rewards are based on the similarity between the LLM's predictions and the outputs from the action/history compression heads. In future work with richer environments, we anticipate using more sophisticated methods for estimating the similarity

Algorithm 1 iLLM(hop) Algorithm

Require: Interaction budget N, Horizon T, Policy π, Large language model LLM, Action generation head f^a, History compression head f^{hc}, Environment env

for $i \leftarrow 1$ to N **do**
    $o_1 \leftarrow$ env.reset()
    for $t \leftarrow 1$ to T **do**
        $x_t \leftarrow \mathbf{E}^\top \mathrm{softmax}(\beta \mathbf{EP}(o_t \cdot a_{t-1}))$ {Translated representation}
        $Z_h \leftarrow LLM(c_{t-1}, x_t)$ {History representation}
        $Z_p^{'} \leftarrow$ get_action_prompt() {Action generation prompt}
        $Z_p^{''} \leftarrow$ get_history_prompt() {History compression prompt}
        $\bar{a}_t \leftarrow \max_{a^i \in \{1,\ldots,K\}} LLM(a^i | Z_h, Z_p^{'})$ {Get next action from the LLM}
        $\widetilde{s_t} \leftarrow \sigma(LLM(Z_h, Z_p^{''}))$ {Get mean-pooled representation of generated summary}
        Call policy $\pi(CNN(o_t) \cdot Z_h)$ {Sample policy and get outputs of the two heads f^a, f^{hc}}
        Compute intrinsic rewards r_t^a and r_t^{hc} based on head outputs, and $\bar{a}_t$ and $\widetilde{s_t}$, respectively {Intrinsic Rewards}
        Store $r_t^* = r_t + b_t = r_t + r_t^a + r_t^{hc}$ along with standard experience in optimization batch B_i {Store agent's experience}
    end for
    Update(π) using B_i {Update the policy}
end for=0

between the agent's trajectory and those predictions, such as using a CLIP-based objective Yuan et al. (2023).

A promising research direction that emerges from our work is the refinement of the mapping process between state-action pairs and the LLM's internal representation. While iLLM successfully harnesses Hopfield networks to align them, there is room for further optimization. Advancements in multimodal techniques for semantically rich text encoding and decoding may play an important role in enhancing the agent's capacity to interpret and respond to *translated* token inputs Wang et al. (2023). We also plan to incorporate modern Hopfield networks to further mitigate the alignment gap Fürst et al. (2022).

Finally, *action generation* and *history compression* tasks are less helpful in domains where human common sense is irrelevant or cannot be expressed in language (e.g., fine-grained manipulation), or action information is not naturally encoded as a language string. An avenue for future work is prompt engineering in order to craft more suitable language tasks.

Appendix B. Implementation Details

As stated previously, PPO was used as our policy learning method. We choose AdamW Loshchilov and Hutter (2019) as optimizer for all environments with default values for weight decay, and clip the norm of the gradients to 0.5. In all experiments, we utilized the 4-bit quantized version of Transfo-XL 280M Dai et al. (2019). Default values were kept for all parameters, except temperature, which was set to 0 to ensure deterministic responses. This language model was employed to compute the embeddings of state-action descriptions Z_h. Similarly, we embedded the action generation and summarization prompts Z_p with the same embedding model. To accelerate the LLM inference, we utilized Lamorel lam (2023) and caching for the action generation and history compression tasks. Furthermore, querying of the LLM for the two language tasks took place in-batch, following the completion of each rollout. Note that due to image-based tasks, we could not report the results of Pangu and PAE in some experiments.

In iLLM(obs), the policy takes as input the current observation encoded through a CNN. In iLLM(hop), the current observation is encoded with a CNN and concatenated with Z_h to form the input for the policy. As mentioned above, the Hopfield module is specifically designed to handle multimodal inputs, such as visual observations, textual descriptions, and numerical data. By translating these diverse inputs into a unified token embedding space, the Hopfield module allows the LLM to process and integrate information from multiple modalities effectively.

f^a and f^{hc} are parametrized by two fully connected layers with hidden dimensions 1024. To maintain a consistent scale of the intrinsic rewards, it is useful to normalize them. This can be achieved by dividing the intrinsic rewards by a running estimate of the standard deviations of the sum of discounted intrinsic rewards. We set $\lambda = 0.8$ and $\beta = 0.5$ to weight the intrinsic rewards. The simplified pseudo-code demonstrating the training procedure of iLLM(hop) is depicted in Alg 1, where, for the sake of clarity, we purposely show iLLM querying the LLM after each interaction with the environment. Note that action generation and history compression queries (lines 8-9) can be performed in-batch following the competition of each rollout to reduce the computational cost.

A graphical illustration of iLLM is provided in Figure 1 of the main manuscript. First, the LLM first retrieves a representation of the recent *(action,observation)* pairs, which is aligned with its internal representation (*input:* $(o_t \cdot a_{t-1})$, *output:* Z_h). If using text-based observations (iLLM(obs)), only the second stage is needed. In this stage, a question prompt Z_p along with the aligned representation Z_h are passed through the LLM in order to obtain either the next action or a summary of the observation (*input:* $[Z_p, Z_h]$, *output:* an action $\bar{a}_t$ or a summary of action-observation pairs). Finally, an intrinsic reward is derived from these next action and summary predictions.

B.1. State-Action History Representation

We now describe the representations of state-action history Z_h that were used in iLLM(obs) and iLLM(hop).

In **iLLM(obs)**, the agent's recent history consists of the last three state-action pairs. The description of the agent's history, Z_h, inputted into the LLM was formatted as follows:

```
Observation 0: {obs_0}
Action 0: {act_0}
Observation 1: {obs_1}
Action 1: {act_1}
Observation 2: {obs_2}
Action 2: {act_2}
```

where **obs** and **act** are the (text-based) descriptions of the observations at time t and actions performed by the agent.

If using the Hopfield module (**iLLM(hop)**), Z_h represents the hidden states from the last hidden layer of the LLM, corresponding to the state-action token. In detail, the input to the Hopfield module consists of flattened grayscaled observations concatenated with the previous action taken (one-hot encoded). The output of the Hopfield module x_t is then passed through the LLM to obtain the state-action token Z_h.

B.2. Action Generation and History Compression Tasks

This section provides details about the action generation and history compression prompts. As described above, Z_p refers to the text embeddings obtained from tokenized text prompts. That is, we employed the base Transfo-XL 280M tokenizer to tokenize the prompts and leveraged the word embeddings of the Transfo-XL model to extract the corresponding embeddings.

The action generation prompt format was set as:

```
You are an expert player playing {task}
Valid actions: {possible actions separated by commas}
You see: {agent history}
Suggest the best action the player can take. Do not recommend
actions that are not possible or not desirable, such as ''Eat door''
. Prioritize actions which involve the object you are facing or
which the agent has not achieved before. What do you do?
```

where **{agent history}** was replaced by Z_h.

The history compression (i.e., summarization) prompt format had the following structure:

```
You are an expert player playing {task}
Recent player's history:: {agent history}
Summarize the main points of the player's history into a short
text:
```

where **{agent history}** was replaced by Z_h. We restricted the number of tokens produced to $L = 64$ in the history compression task.

Appendix C. Environments

C.1. BabyAI-Text

BabyAI-Text is a text-based environment that encapsulates BabyAI, providing a textual description of each observation Chevalier-Boisvert et al. (2018a). We evaluate iLLM on a set of nine tasks in the BabyAI-Text environment Chevalier-Boisvert et al. (2018a). The agent must navigate in procedurally generated rooms that include distractors — useless objects for completing the task.
We selected the following tasks:

- **Key corridor**, a task that requires to pick up an object which is behind a locked door. The key is hidden in another room, and the agent has to explore the environment to find it.
- **Obstructed maze**, a navigation task where a blue ball is hidden in one of the 4 corners of a maze. Doors are locked, doors are obstructed by a ball and keys are hidden in boxes.
- **Go to**, a navigation task that requires reasoning abilities in order to reach the goal object.
- **Pick up**, a navigation task that combines navigation tasks and picking up the object.
- **Put object A next to object B (put next)**, a sequence of 3 tasks, including reaching object A then reaching object B and finally dropping object A next to object B.
- **Open door**, a task that requires inferring that a key is useful for unlocking a door, finding another key, and finally using the toggle action with the key in the door.

A textual description consists of a list of template descriptions with the following structure:

```
"You see a <object> <location>" if the object is a key, a ball, a
box, or a wall.
"You see a(n) open/closed door <location>" , if the agent sees a
door.
"You carry a <object>", if the agent carries an object.
```

where the <object> is composed of an adjective (among six possible colors: blue, red, green, grey, purple, and yellow) and a noun (among four possible: door, key, ball, box). The <location> is given as the number of steps right, left, and or forward from the agent to the object.

C.2. MiniHack

The MiniHack environment based on NetHack (Kuttler et al., 2020) features a larger action space compared to BabyAI, with up to 75 distinct actions. Observations are composed of a 21 $\times$ 79 matrix containing glyph identifiers and a 21-dimensional vector capturing agent statistics such as location and health. Additionally, real natural language messages received during gameplay are included in a 256-dimensional vector termed as a “message”, representing the on-screen display at the screen’s top. Each glyph corresponds to a unique entity, denoted by integers ranging from 0 to 5991. Our study focused on five MiniHack tasks, encompassing navigation challenges like River-Monster and Multiroom-N4-Monster, along with skill acquisition tasks such as LavaCross-Ring, LavaCross-Potion, and LavaCross-Full. The navigation tasks in MiniHack test the agent’s ability to navigate diverse obstacles, from maneuvering boulders to crossing rivers, while skill acquisition tasks exploit NetHack’s vast array of objects, monsters, and dungeon features, exploring their interactions and complexities.

Model	**N=4**	**N=8**	**N=16**
RND	0.59± 0.16	0.51± 0.22	0.09± 0.10
NGU	0.51± 0.20	0.42± 0.25	0.11± 0.13
ELLM	0.78± 0.03	0.66± 0.01	0.64± 0.05
APT	0.57± 0.19	0.48± 0.17	0.44± 0.20
ChibiT	0.56± 0.23	0.51± 0.15	0.48± 0.17
iLLM(obs)	**0.96± 0.01**	**0.94± 0.00**	0.91 ± 0.02
iLLM(hop)	**0.96± 0.00**	0.92± 0.01	**0.94 ± 0.01**

Table 1: Success rate for the agents on the *Go To* task for different number of distractors ($N \in \{4, 8, 16\}$). The success rate is provided over 10 seeds with standard deviation after 100M training steps.

C.3. Crafter

Crafter is an open-ended environment in which exploration is required to discover long-term survival strategies Hafner (2021). It is a 2D variant inspired by Minecraft, featuring a procedurally generated and partially observable world world. Crafter enables collecting and creating a set of artifacts organized along an achievement tree, which lists all possible achievements and their respective prerequisites. Despite lacking a single main task, tracking the agent's advancements along the achievement tree provides insights into its progress within Crafter.

Appendix D. Ablation Results

D.1. Impact of the Number of Distractors

This section aims to assess how much distractors impact the proposed method. These evaluations were performed in an environment with one room on the *Go To* task (BabyAI-Text). We report in Table 1 the average success rate for 4, 8, and 16 distractors. RND and NGU significantly degrade as the number of distractors increases, with a success rate decreasing by ≈65% from 4 to 16 distractors. We also observe a slight performance loss in iLLM agents when the number of distractors is larger than 8. A potential rationale behind this phenomenon is that the LLM effectively directs the iLLM agent's attention to the relevant aspects of the environment, rapidly discarding distractors and noise in the observations. On the other hand, model-based curiosity is more impacted by distractors as it favors a full exploration of the state space, resulting in the exploration of a larger number of irrelevant behaviors. Overall, the present algorithm appears to be reasonably robust to distractors.

D.2. Language Model Tasks

As described in section 3, iLLM relies on action generation and history compression tasks. In this experiment, we evaluate the performance of the proposed method with two other

Table 2: Final mean performance (± std) of iLLM(hop) trained with different language tasks, including action generation (♣), history compression (♥), goal generation (♠), and plan generation (♦). For the sake of generality, we report results where the recent history Z_h consists of translated state-action pairs, iLLM(hop). Averages over 10 runs.

	BabyAI		Atari		MiniHack	
Method	GoToObj	PutNextLocal	MR	PrivateEye	LavaCross-Full	River-Monster
iLLM + ♣	0.85±0.01	0.43±0.09	2,118±329	3,981±420	0.91±0.01	0.35±0.09
iLLM + ♥	0.90±0.01	0.41±0.10	2,456±198	3,565±454	0.98±0.03	0.35±0.07
iLLM + ♠	0.85±0.02	0.42±0.08	2,024±176	3,429±500	0.96±0.02	0.32±0.10
iLLM + ♦	0.71±0.07	0.12±0.13	1,365±202	2,560±321	0.86±0.09	0.26±0.06
iLLM + ♣ + ♥	0.92±0.01	0.49±0.12	2,632±277	4,422±376	0.98±0.03	0.38±0.12
iLLM + ♠ + ♦	0.83±0.05	0.42±0.06	2,139±251	2,945±400	0.89±0.07	0.30±0.08
iLLM + ♣ + ♦	0.84±0.03	0.45±0.11	2,299±244	3,647±178	0.91±0.01	0.31±0.05
iLLM + ♣ + ♠	0.93±0.02	0.46±0.05	2,432±199	4,295±420	0.95±0.01	0.34±0.06
iLLM + ♥ + ♠	0.93±0.01	0.48±0.09	2,312±271	3,999±392	0.98±0.02	0.36±0.08
iLLM + ♥ + ♦	0.86±0.07	0.39±0.08	2,202±302	2,876±287	0.96±0.01	0.31±0.06
iLLM + ♣ + ♥ + ♠	0.93±0.03	0.51±0.08	2,553±298	4,500±356	0.97±0.02	0.39±0.10
iLLM + ♣ + ♥ + ♦	0.89±0.02	0.47±0.08	2,421±230	4,053±312	0.98±0.03	0.36±0.09
iLLM + ♠ + ♥ + ♦	0.88±0.04	0.45±0.12	2,376±255	3,971±253	0.96±0.01	0.34±0.09
iLLM + ♣ + ♠ + ♦	0.90±0.03	0.50±0.06	2,112±303	4,421±443	0.94±0.02	0.34±0.07
iLLM + ♣ + ♠ + ♥ + ♦	0.93±0.02	0.50±0.05	2,510±300	4,499±398	0.96±0.02	0.37±0.05

language tasks, including goal generation and plan generation. For **goal generation**, we queried the LLM with the following instruction:

```
You are an expert player playing {task}
Recent player's history: {agent history}
Suggest the next goal to reach based on the things you see and
previous actions. A goal should either be a single valid word or
a phrase. Only make suggestions that are reasonable given the
current scene (e.g. only ``Open door'' if a door is visible).
Prioritize goals that involve the object you are facing or that
the agent has not achieved before.
```

and for **plan generation**:

```
You are an expert player playing {task}
Recent player's history: {agent history}
Suggest the best sequence of actions the player can take. An
action should either be a single valid word or a phrase. Only
make suggestions that are reasonable given the current scene
(e.g., only ``Open door'' if a door is visible). Prioritize
actions which involve the object you are facing or which the
agent has not achieved before. What do you do (include 2-7
actions)?
```

The associated heads were trained in the same fashion as the history compression head. As shown in Table 2, agents leveraging action generation and history compression tasks demonstrate reasonable scores on all tasks. Notably, *goal generation* achieves high performance on BabyAI-text, but the average return decreases on more complex games such as

Method	MR	PrivateEye	Gravitar	Pitfall	Seaquest
PPO	11±4	0.0±0.0	120±14	-7±2	1,245±199
iLLM(hop)	**2,632±277**	**4,422±376**	**4,044±559**	**125±24**	**18,851±2,930**
iLLM(hop)(no reward)	1,452±201	2,871±265	2,450±287	3±5	15,888±3,012

Table 3: Performance of iLLM that solely distills an LLM via the action generation and history compression heads on Atari tasks. Under this setting, iLLM(hop)(no reward) does not receive any intrinsic rewards. All methods are tested with 10 random seeds. Averages over 10 runs for 100 million training steps.

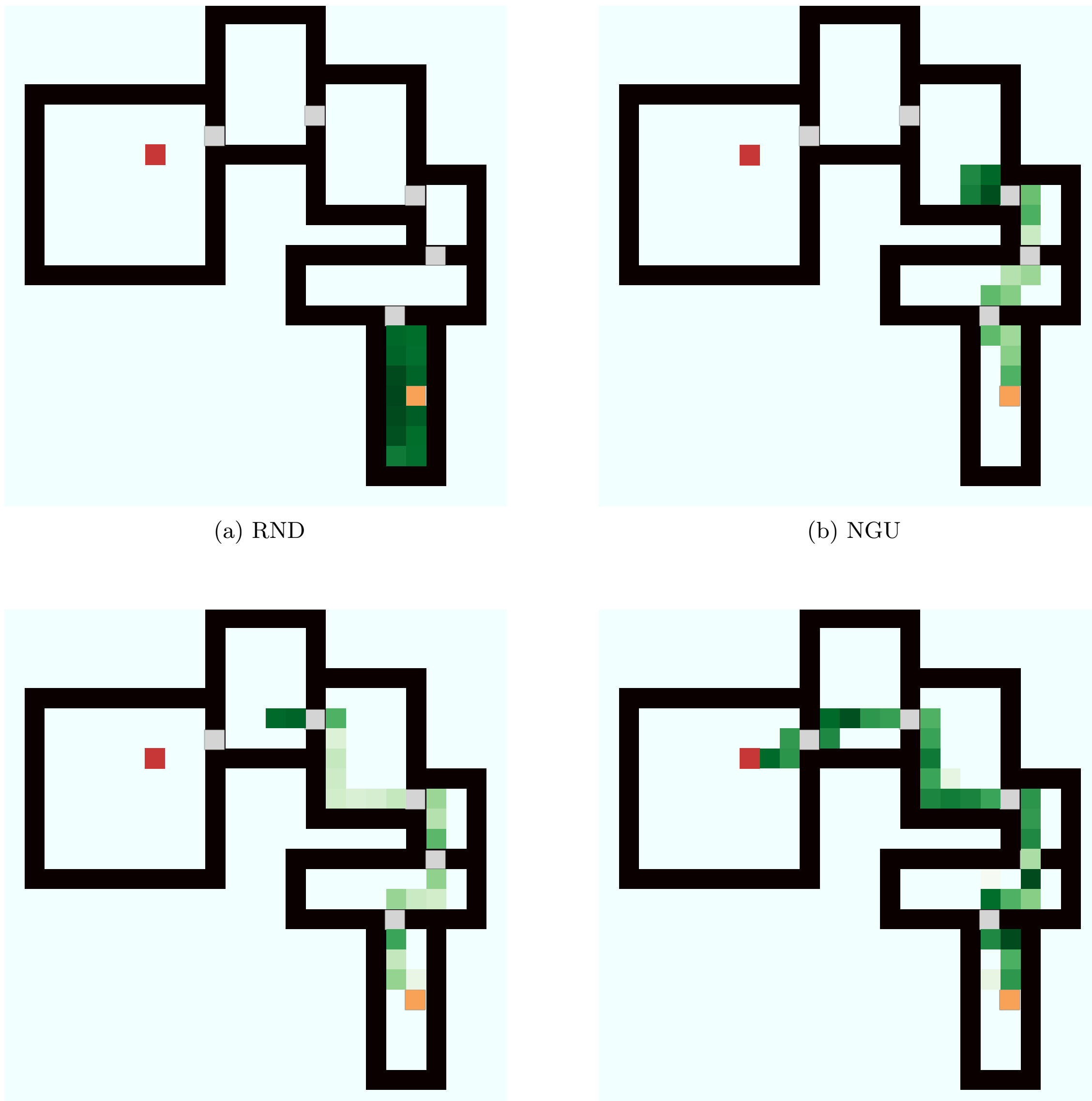
Atari. Besides, we find that the model performance saturates when the number of language tasks goes larger. Namely, leveraging more language tasks cannot monotonically promote performance when the number is larger than 2. Therefore, we empirically set the number of tasks to 2 by default. In addition, it is evident that *history compression* is the reward that accelerates the most exploration. In contrast, both *action generation* and *goal* generation result in slightly lower improvements. Finally, *plan generation* proved to be the least effective reward, which may be attributed to the difficulty of generating meaningful plans with a small language model like Transfo-XL.

D.3. State Visitation

To test the good exploration coverage of our approach, we trained iLLM(hop) on a procedurally-generated environment: MiniGrid-MultiRoom-N6-v0 Chevalier-Boisvert et al. (2018b). Utilizing MiniGrid-MultiRoom-N6-v0, which involves sequential visits to multiple rooms, facilitates a clearer measurement of the exploration progress. Fig. 1 demonstrates that in order to remain curious, the agent is pushed to explore distant regions of the state space, which entails that its coverage increases over time. Experimental results highlight that in procedurally-generated tasks, iLLM provides enough exploration incentive for learning useful behaviors. That is, in order to remain curious, agents are pushed to explore diverse behaviors, enabling the discovery of new rooms and interactions with the objects. Unlike RND and NGU, which spend time to exploring local behaviors that have a low interest, iLLM(hop) rapidly drives the agent towards the goal being pursued (red cell).

D.4. Diverse Exploration

In this set of experiments, we aim to evaluate the contribution of distilling an LLM to the efficacy of our proposed method. One of our assumptions is that appending f^a and f^{hc} to the policy and training them to match the pretrained LLM's outputs enables iLLM to leverage world knowledge through their respective gradients. In order to validate this hypothesis, we report the results of PPO, iLLM(hop), and iLLM(hop)(no reward) in the five Atari games. The latter method was trained without access to the intrinsic rewards. Thus, LLM's prior knowledge is only distilled through gradients of f^a and f^{hc} — the agent does not receive an explicit incentive to explore. As indicated in Table 3, in the five tasks, we find that distilling an LLM, as done by *iLLM(hop)(no reward)*, leads to higher performance compared to plain PPO. This highlights that even in the absence of any intrinsic rewards, the



(a) RND (b) NGU (c) ELLM (d) iLLM(hop)

Figure 1: State visitation heatmaps over 10 runs for RND, NGU, ELLM, and iLLM(hop) on a random environment from the MiniGrid-MultiRoom-N6-v0 task. The initial agent's location is denoted with an orange cell, while the target is denoted with a red cell. We trained the models for 40m frames.

Method	MR	PrivateEye	Gravitar	Pitfall	Seaquest
iLLM(hop)	**2,632±277**	**4,422±376**	**4,044±559**	**125±24**	**18,851±2,930**
iLLM(hop) (ϱ = 0.50)	1,989±134	3,522±398	3,232±356	65±29	15,437±2,273
iLLM(hop) (ϱ = 1.0)	2,012±209	3,408±401	3,018±410	59±31	13,419±2,414

Table 4: LLM stability study. We report the average reward in randomized-versions of Atari games. All methods are tested with 10 random seeds. Averages over 10 runs for 100 million training steps.

two prediction heads attached to the policy provide sufficient domain-specific priors to speed up the training process. We also see here that iLLM(hop) is more competitive, suggesting that including intrinsic rewards is better suited for exploration than agents without access to intrinsic rewards.

D.5. LLM Stability

One potential drawback of leveraging LLMs is their stability when facing various input conditions. As mentioned above, in our experiments, we set the temperature to 0 in order to enhance stability. In addition, we observed that limiting the number of tokens produced to $L = 64$ in the history compression task makes predictions more stable. It is also important to note that the LLM in our approach is used to encourage exploration through a small reward incentive, rather than serving as an oracle providing plans or next actions. This makes iLLM inherently more stable than approaches that rely on the LLM for direct decision-making.

To further validate the stability of iLLM, we study the effect of adding perturbations to the observations. Namely, with a probability $\varrho \in \{0.50, 1.0\}$, a noise pattern (32×32) is displayed on the lower right of the observation - TV screen. The noise is sampled from [0,255] independently for each pixel. As reported in Table 4, the performance iLLM deteriorates due to the stochasticity. Nevertheless, our approach is reasonably robust to randomized observations. As iLLM does not directly relies on the LLM's output but is driven by a proxy reward obtained from the LLM, iLLM remains stable under various input conditions. Overall, LLM stability does not appear to be a concern for iLLM under our experimental design choices, but we leave it to future work to explore this direction further.

D.6. Impact of the Dimension of the Action Space

This experiment aims to assess the sensitivity of the present reward to the size of the action space by implementing three action space settings on the *Go to* task:

- **Original**: the action space consists of the only 3 useful actions: turn left, turn right, go forward.

- **Augmented**: the action space consists of the 6 actions that can be performed in the environment. The agent can select one action among 3 useful and 3 useless actions that are pick up, drop and, toggle.

- **Irrelevant**: the action space consists of 9 actions (3 useful and 6 useless with pick up, drop, toggle, sleep, do nothing and think). The last three actions have been selected

Model	Original	Augmented	Irrelevant
RND	0.51± 0.22	0.42± 0.17	0.27± 0.08
NGU	0.42± 0.25	0.37± 0.21	0.25± 0.09
ELLM	0.66± 0.01	0.60± 0.05	0.55± 0.03
APT	0.48± 0.17	0.46± 0.18	0.42± 0.09
ChibiT	0.56± 0.23	0.51± 0.25	0.45± 0.12
iLLM(obs)	**0.94± 0.00**	0.88± 0.08	0.81± 0.04
iLLM(hop)	0.92± 0.01	**0.89± 0.11**	**0.83± 0.09**

Table 5: Effect of using different action space sizes on iLLM performance in the *Go to* task. We report results for three settings: original (3 useful actions), augmented (6 actions), and irrelevant (9 actions). Results are averaged over 10 trials.

such that they are irrelevant for solving the *Go To* task and therefore should not impact an agent that has knowledge about the world.

In Table 5, we present the evaluation conducted in an environment comprising 1 room and 8 distractors. The result of this experiment is that using *augmented* or *irrelevant* actions does not significantly degrade the performance of the proposed approach, while the average reward of other agents decreases with the exception of APT. Upon looking at the videos, we observed that other agents tend to frequently select useless and irrelevant actions. On the other hand, from the onset of the training, our agents are rewarded for trying useful actions first, which allows us to achieve higher sample efficiency.

D.7. Use of LLM

At each time step the LLM is first employed to retrieve a representation of the (action, observation) pair, which is aligned with its internal representation (input: $(o_t \cdot a_{t-1})$, output: Z_h). Then, a prompt Z_p along with the aligned representation Z_h are passed through the LLM in order to obtain either the next action or a summary of the observation (input: Z_p, Z_h, output: an action $\bar{a}_t$ or a summary of the action-observation pair).

To illustrate the use of the LLM in our framework, Figure 2 provides an example specific to the action generation task. Although we focus on action generation here for brevity, it is important to note that task 1 (alignment) is performed only once per time step. The same aligned representation, Z_h, is then utilized for both the action generation and history compression tasks. For simplicity, we assume that Z_h is retrieved solely from the most recent state-action pair.

D.8. Choice of Foundation Model

We now seek to evaluate the performance of our methodology using various foundation models on Atari games. Specifically, we compare the results obtained by employing Transfo-XL 280M, GPT2 137M, GPT2 870M, Llama2-7b, and Llama-3 8b. The results, presented in Table 6, demonstrate that the performance of iLLM is generally robust across different foundation models. While LLama3 exhibits a significantly higher average return (t-test

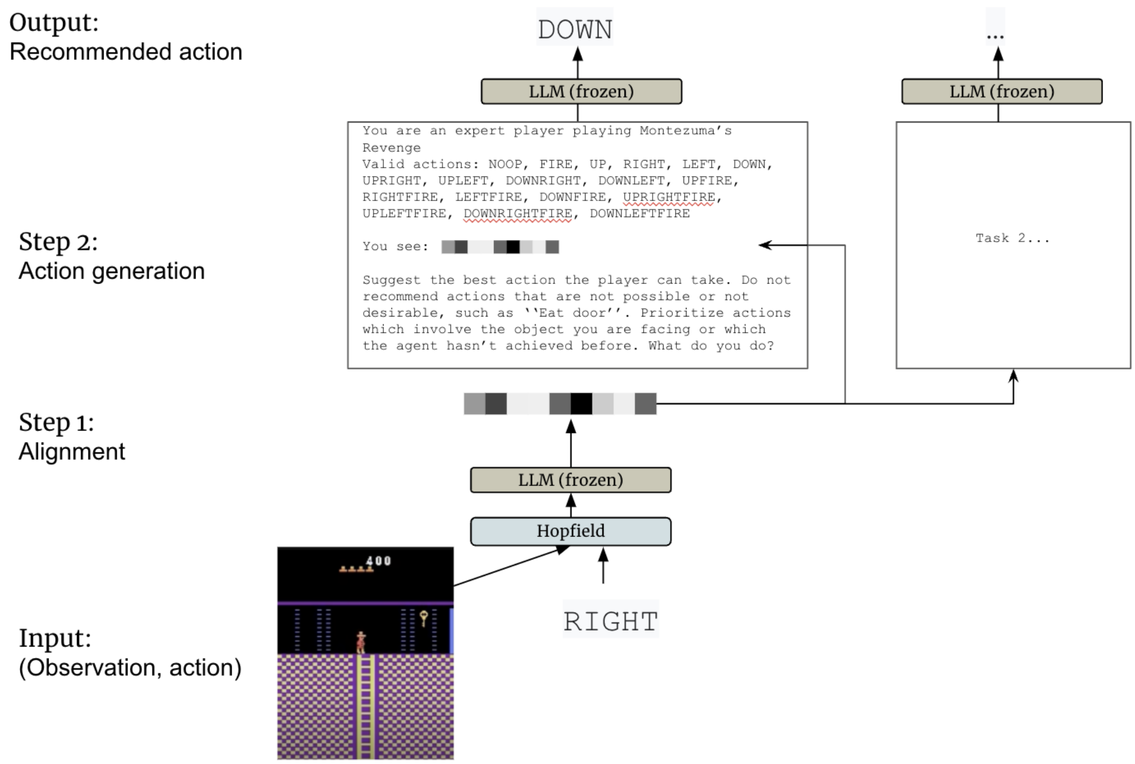


Figure 2: Example of LLM usage in iLLM for the action generation task. iLLM first aligns a (action, observation) pair with the internal representation of the LLM. Then, a prompt along with the aligned representation are passed through the LLM in order to obtain the next action.

Method	MR	PrivateEye	Gravitar	Pitfall	Seaquest
iLLM (hop · Transfo-XL)	2,632±277	4,422±376	4,044±559	125±24	18,851±2,930
iLLM (hop · GPT2 137)	2,299±256	4,134±350	3,756±498	98±31	16,648±2,615
iLLM (hop · GPT2 870)	2,524±312	4,288±321	3,877±522	110±22	17,461±2,646
iLLM (hop · Llama2)	2,953±269	4,954±341	4,644±488	176±31	20,129±2,424
iLLM (hop · Llama3)	**3,220±312**	**5,098±430**	**4,831±502**	**277±55**	**23,980±2,731**

Table 6: Performance of iLLM with different types of foundation LLMs on Atari tasks. All methods are tested with 10 random seeds. Averages over 10 runs for 100 million steps.

$p < 0.05$), Transfo-XL achieves similar performance but with a much lower inference time. Namely, Transfo-XL model has only 280M parameters compared to the 8B parameters of LLama3.

References

Lamorel. https://github.com/flowersteam/lamorel, 2023.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. *arXiv preprint arXiv:1810.08272*, 2018a.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid, 2018b.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. Guiding pretraining in reinforcement learning with large language models. *arXiv preprint arXiv:2302.06692*, 2023.

Andreas Fürst, Elisabeth Rumetshofer, Johannes Lehner, Viet T Tran, Fei Tang, Hubert Ramsauer, David Kreil, Michael Kopp, Günter Klambauer, Angela Bitto, et al. Cloob: Modern hopfield networks with infoloob outperform clip. *Advances in neural information processing systems*, 35:20450–20468, 2022.

Danijar Hafner. Benchmarking the spectrum of agent capabilities. *arXiv preprint arXiv:2109.06780*, 2021.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 6292–6299, 2018.

Emilio Parisotto and Ruslan Salakhutdinov. Efficient transformers in reinforcement learning using actor-learner distillation. *arXiv preprint arXiv:2104.01655*, 2021.

Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*, 2023.

Benjamin Spector and Chris Re. Accelerating llm inference with staged speculative decoding. *arXiv preprint arXiv:2308.04623*, 2023.

Wenhai Wang, Zhe Chen, Xiaokang Chen, Jiannan Wu, Xizhou Zhu, Gang Zeng, Ping Luo, Tong Lu, Jie Zhou, Yu Qiao, et al. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. *arXiv preprint arXiv:2305.11175*, 2023.

Xingrui Yu, Yueming Lyu, and Ivor Tsang. Intrinsic reward driven imitation learning via generative model. In *International conference on machine learning*, pages 10925–10935. PMLR, 2020.

Haoqi Yuan, Chi Zhang, Hongcheng Wang, Feiyang Xie, Penglin Cai, Hao Dong, and Zongqing Lu. Skill reinforcement learning and planning for open-world long-horizon tasks. *arXiv preprint arXiv:2303.16563*, 2023.