

From Actions to Words: Towards Abstractive-Textual Policy Summarization in RL

Sahar Admoni
Technion – IIT
Haifa, Israel
saharad@campus.technion.ac.il

Assaf Hallak
Nvidia Research
Tel Aviv, Israel
ahallak@nvidia.com

Yftah Ziser
University of Groningen, Nvidia
Research
Tel Aviv, Israel
yziser@nvidia.com

Omer Ben-Porat
Technion – IIT
Haifa, Israel
omerbp@technion.ac.il

Ofra Amir
Technion – IIT
Haifa, Israel
oamir@technion.ac.il

ABSTRACT

Explaining reinforcement learning agents is challenging because policies emerge from complex reward structures and neural representations that are difficult for humans to interpret. Existing approaches often rely on curated demonstrations that expose local behaviors but provide limited insight into an agent’s global strategy, leaving users to infer intent from raw observations. We propose *SySLLM* (Synthesized Summary using Large Language Models), a framework that reframes policy interpretation as a language-generation problem. Instead of visual demonstrations, *SySLLM* converts spatiotemporal trajectories into structured text and prompts an LLM to generate coherent summaries describing the agent’s goals, exploration style, and decision patterns. *SySLLM* scales to long-horizon, semantically rich environments without task-specific fine-tuning, leveraging LLM world knowledge and compositional reasoning to capture latent behavioral structure across policies. Expert evaluations show strong alignment with human analyses, and a large-scale user study found that 75.5% of participants preferred *SySLLM* summaries over state-of-the-art demonstration-based explanations. Together, these results position *abstractive textual* summarization as a paradigm for interpreting complex RL behavior.¹

KEYWORDS

Reinforcement Learning, Policy Summarization, Explainable AI

ACM Reference Format:

Sahar Admoni, Assaf Hallak, Yftah Ziser, Omer Ben-Porat, and Ofra Amir. 2026. From Actions to Words: Towards Abstractive-Textual Policy Summarization in RL. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 9 pages. <https://doi.org/10.65109/EGVR4787>

¹ Code available at <https://github.com/saharad1/SySLLM>. Supplementary material (including the appendix) is available at <https://arxiv.org/abs/2503.10509>.



This work is licensed under a Creative Commons Attribution International 4.0 License.

1 INTRODUCTION

Reinforcement learning (RL) agents are increasingly deployed in sequential decision-making domains, yet their policies remain opaque to human stakeholders. This opacity limits trust, adoption, and effective debugging. While many explainable reinforcement learning (XRL) methods provide *local* insight into individual states or decisions, our focus is on *global* explanations that aim to capture an agent’s strategy across trajectories. Existing paradigms face trade-offs between *expressiveness*, *scalability*, and *faithfulness*. Extractive methods such as saliency maps [7, 16, 36, 38] and demonstration-based policy summaries [4, 12, 15, 27] capture only fragments of behavior. Symbolic approaches such as rules or decision trees [30, 34, 35, 40] often collapse in high-dimensional or partially observed environments. Across these approaches, a core flaw persists: they transform observations but do not synthesize them into holistic accounts, leaving users to reconstruct intent, adaptability, and failure modes from fragmented evidence [5, 6].

Large language models (LLMs) appear to offer a promising alternative. Their capacity for abstraction, compositional reasoning, and natural language generation [44] suggests that they could distill trajectories into human-readable accounts of agent behavior. However, applying them to RL policies is far from straightforward. LLMs are trained on static text-based corpora, while RL agents generate dynamic spatio-temporal trajectories grounded in states, actions, and rewards [26]. This mismatch in modality makes faithful policy summarization non-trivial. Specifically, naive application risks producing fluent and plausibly sounding behavior descriptions that do not accurately reflect the underlying policy.

To address this challenge, we propose a *textual-abstractive paradigm* for policy summarization, in which explanations are expressed as natural language narratives integrating evidence from a database of policy execution traces, which we term the textual experience buffer (TEB). Unlike extractive or symbolic methods, this paradigm explicitly targets policy-level regularities and behavioral motifs. We formalize the task as a mapping from an agent’s experience buffer to a textual summary, guided by the desiderata of *expressiveness*, *scalability*, and *faithfulness*. To realize these principles in practice, we adopt a conceptual optimization view in which summaries balance *coverage*, *parsimony*, and *fidelity*. This formulation grounds both our methodology and evaluation, ensuring that system design

choices are principled and directly address the unique challenges of applying LLMs to RL.

Building on this formulation, we introduce **SySLLM**, a framework that leverages LLMs to generate global policy summaries. SySLLM operates in two stages: agent–environment trajectories are first transformed into structured natural language descriptions of observations and actions, which are then synthesized into higher-level accounts through carefully designed prompting. To scale beyond context window limitations, SySLLM performs hierarchical summarization over large buffers. To mitigate variability in LLM outputs, it generates multiple candidate summaries and aggregates them into a consensus using embedding-based similarity. Together, these mechanisms ensure that SySLLM produces summaries that are general enough to capture policy-level regularities while specific enough to reflect distinctive behaviors.

We evaluated SySLLM across five MiniGrid environments and the Crafter domain, covering nine qualitatively distinct agent policies. Expert evaluation shows a strong alignment between SySLLM summaries and expert summaries, achieving high recall and precision scores that reflect faithful coverage of expert-identified behaviors. A user study with 192 participants further demonstrates that users strongly prefer textual summaries over demonstration-based summaries such as HIGHLIGHTS-DIV [3], while performing equally well or better on policy identification tasks.

Our main contributions are threefold: (1) we introduce and formalize the task of *abstractive-textual policy summarization* in RL, framing it as a mapping from an agent’s experience buffer to natural language narratives defined by desiderata of expressiveness, scalability, and faithfulness; (2) we present **SySLLM**, a framework that leverages LLMs to synthesize structured trajectory descriptions into coherent global summaries, incorporating hierarchical summarization and consensus aggregation to address long horizons and variability in outputs, and (3) we provide extensive empirical validation through expert evaluations and a large-scale user study, demonstrating that SySLLM produces faithful summaries that are strongly preferred by participants over state-of-the-art demonstration-based baselines.

2 RELATED WORK

Prior work in XRL spans both local explanations of individual decisions and global summaries of agent behavior, with the shared goal of improving policy interpretability for humans [31]. Saliency and visualization methods highlight influential inputs [16, 22] but are often local and fragile; demonstration-based methods summarize behavior via selected trajectories [4, 12] but place interpretive burden on users; surrogate models distill policies into rules or decision trees [8, 11], though with fidelity–scalability trade-offs; and causal or reward-based explanations [24, 28] provide structured insights but typically require access to internals or domain expertise. Collectively, these techniques emphasize fragments or simplified proxies rather than synthesizing global accounts of an agent’s strategy.

Large language models (LLMs) bring complementary capabilities of abstraction, reasoning, and fluent text generation [9]. While prior work mainly uses LLMs during training—for example, guiding exploration or constructing world models [13, 17]—their potential as *explanation generators* has received less attention. Some studies

prompt LLMs to narrate behavior in real time [41, 43] or to build symbolic simulators [2], but these efforts are often ungrounded in actual dynamics and focus on local rather than global explanations.

Natural language explanations for RL agents have also been studied. The early template-based methods [20, 35] prioritized accessibility, but were brittle, while neural rationalization approaches translated trajectories into free form text [14, 29]. These provide interpretability but generally rely on handcrafted structures, focus on local justifications, or lack scalability across diverse scenarios. They rarely capture holistic behavioral patterns or leverage broader common sense knowledge. These limitations motivate our textual-abstractive paradigm, where LLMs synthesize coherent global summaries of agent behavior.

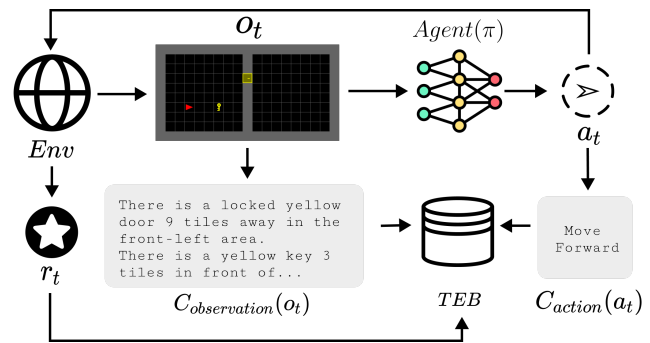


Figure 1: Collecting the textual experience buffer (Section 4.2).

3 PROBLEM FORMULATION

In this section, we formalize the abstractive-summarization problem. Specifically, we describe the setting, define the summarization task, and introduce conceptual principles that guide both our methodology and our evaluation.

3.1 Setting

We consider an RL environment modeled as a partially observable Markov decision process (POMDP):

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, T, O, R, \gamma \rangle,$$

where \mathcal{S} is the state space, \mathcal{A} the action space, \mathcal{O} the observation space, $T(s' | s, a)$ the transition kernel, $O(o | s, a)$ the observation function, $R(s, a)$ the reward function, and $\gamma \in (0, 1]$ the discount factor.

An agent follows a stochastic policy

$$\pi : \mathcal{O} \rightarrow \Delta(\mathcal{A}),$$

which maps each observation $o \in \mathcal{O}$ to a distribution over actions. The interaction induces a distribution over trajectories

$$\tau = (o_0, a_0, r_0, \dots, o_T), \quad \tau \sim \pi, T, O.$$

For summarization, we assume access not only to isolated trajectories but to an *experience buffer*

$$\mathcal{B}_\pi = \{\tau_1, \dots, \tau_N\},$$

which aggregates multiple episodes sampled from π . This buffer serves as the raw material based on which textual explanations are generated.

3.2 Policy Summarization Task

We define a *policy summarizer* as a mapping

$$f : \mathcal{B}_\pi \rightarrow \mathcal{T},$$

where \mathcal{T} denotes the space of abstractive textual explanations.

We require f to approximate three key principles. **Expressiveness**: capture recurring behavioral patterns (e.g. “the agent prioritizes unlocking doors before exploring rooms”) rather than isolated actions. **Scalability**: operate over long horizons and large buffers while maintaining concise summaries. **Faithfulness**: reflect the actual distribution of behaviors under π , avoiding hallucinated or spurious strategies.

Conceptual Optimization View. We frame summarization as a conceptual optimization problem:

$$T^* = \arg \max_{T \in \mathcal{T}} \mathcal{U}(T | \mathcal{B}_\pi),$$

where \mathcal{U} is a utility function balancing: *coverage* (operationalizing expressiveness by accounting for recurring behaviors), *parsimony* (operationalizing scalability through concise abstraction), and *fidelity* (operationalizing faithfulness via alignment with empirical evidence in \mathcal{B}_π).

This formulation is not solved directly. Instead, it provides a guiding lens for both *methodology* and *evaluation*: SySLLM instantiates coverage through multi-trajectory aggregation and hierarchical summarization, parsimony through representative summary selection, and fidelity through expert alignment and user validation. In the following sections, we detail how these principles are operationalized in practice and evaluated empirically.

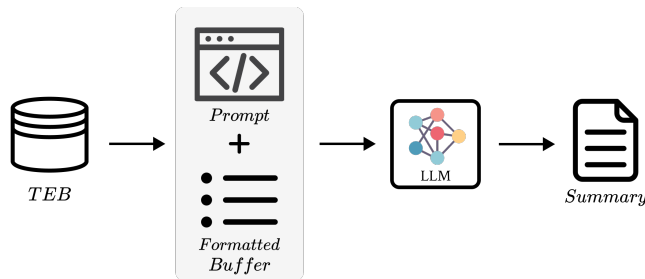


Figure 2: Generating global policy summaries (Section 4.3).

4 SySLLM FRAMEWORK

Our SySLLM (Synthesized Summary using LLMs) framework formalizes policy summarization as a two-phase process: (i) *experience collection and captioning*, which transforms trajectories into natural language traces stored in a *Textual Experience Buffer (TEB)*, and (ii) *abstractive summarization*, which synthesizes the TEB into a concise global description of the agent’s policy using a large language model (LLM). Algorithms 1–2 specify the pipeline, while Figures 1–2 illustrate its two phases.

4.1 Captioners

The first phase converts raw trajectories into textualized experiences. For a trajectory $\tau = (o_1, a_1, r_1, \dots, o_T)$, we define two captioning functions:

$$C_{\text{obs}} : \mathcal{O} \rightarrow \Sigma^*, \quad C_{\text{act}} : \mathcal{A} \rightarrow \Sigma^*,$$

where Σ^* denotes the set of natural language strings. At each step t , the pair (o_t, a_t) is mapped to $(C_{\text{obs}}(o_t), C_{\text{act}}(a_t))$, yielding a textual description of the observation and the action taken.

Observation Captioner. C_{obs} produces structured descriptions of salient percepts (e.g. “The agent is facing a locked door with a key to the left.”).

Action Captioner. C_{act} verbalizes the agent’s action (e.g. “move forward,” “pick up the key.”).

This follows prior work on language rounding [23, 32]. In practice, captioners may be rule-based, vision-language based, or hybrid; the concrete instantiation is left as a domain-specific choice and is discussed further in Section 9.

4.2 Constructing the Textual Experience Buffer

From the N sampled episodes, we construct a *Textual Experience Buffer (TEB)*:

$$TEB_\pi = \langle e_1, e_2, \dots, e_L \rangle, \quad e_t = \langle \hat{o}_t, \hat{a}_t, r_t, epID \rangle,$$

where $\hat{o}_t = C_{\text{obs}}(o_t)$ and $\hat{a}_t = C_{\text{act}}(a_t)$ denote natural language captions of observations and actions. Each tuple records captioned observation, captioned action, reward, and episode identifier, preserving temporal coherence between steps.

Episodes are generated by sampling from the policy π :

$$o_1 \sim \text{RESET}(\text{ENV}), \quad a_t \sim \pi(o_t), \quad (o_{t+1}, r_t, \text{done}) \sim \text{STEP}(a_t),$$

repeating until $\text{done} = \text{True}$. At each step, both observation and action are passed through captioning functions C_{obs} and C_{act} .

The TEB is thus a textual analogue of a replay buffer: it aggregates multiple trajectories in a structured, language-based format that is directly consumable by LLMs, while maintaining the sequential structure necessary for policy-level reasoning. Further details on the fields stored in the TEB is provided in the appendix.

4.3 Abstractive Summarization

Let \mathcal{B} denote the space of textual experience buffers, where each $TEB_\pi \in \mathcal{B}$ is a finite ordered sequence of captioned experience tuples collected from executions of policy π .

The second phase maps the TEB to a global narrative $T \in \mathcal{T}$. We define the summarizer as

$$f_\theta : \mathcal{B} \rightarrow \mathcal{T}, \quad T = f_\theta(TEB_\pi),$$

where Σ^* denotes the space of finite token sequences and f_θ is instantiated by an LLM conditioned on a structured prompt.

Prompt Construction. The prompt follows a hierarchical structure, inspired by Chain-of-Thought reasoning [25, 42], which decomposes complex behavioral analysis into specific components. This design guides the LLM to progressively move from low-level traces in the TEB to higher-level abstractions of the policy. The prompt consists of:

- (1) **General Instructions:** define the summarization task.

- (2) **Environment Context:** describe task objectives and constraints.
- (3) **Textual Experience Buffer:** provide TEB in structured form to preserve sequentiality.
- (4) **Output Specification:** constrain the output to a concise summary in natural language.

Prompt design details are provided in the appendix.

Scalability via Hierarchical Summarization. A fundamental challenge in LLM-based summarization is the bounded context size: The TEB may exceed the maximum token budget κ of the model. To address this, SySLLM employs a hierarchical procedure that recursively reduces the buffer until it fits within the context window.

Formally, define a summarization operator:

$$\mathcal{S}_\theta : \mathcal{D} \rightarrow \Sigma^*,$$

where \mathcal{D} is any subset of the TEB and Σ^* the space of textual summaries. If $|TEB| \leq \kappa$, we directly apply:

$$T = \mathcal{S}_\theta(TEB).$$

If $|TEB| > \kappa$, we partition the buffer into M disjoint subsets $TEB = \{TEB^{(1)}, \dots, TEB^{(M)}\}$ such that each $|TEB^{(i)}| \leq \kappa$. For each subset we compute intermediate summaries:

$$S_i = \mathcal{S}_\theta(TEB^{(i)}), \quad i = 1, \dots, M,$$

and aggregate them by applying \mathcal{S}_θ again:

$$T = \mathcal{S}_\theta(\{S_1, \dots, S_M\}).$$

This recursive divide-and-conquer scheme:

$$T = \text{HIERARCHICALSUMMARIZE}(TEB, \kappa)$$

ensures that SySLLM remains applicable to arbitrarily large buffers while preserving coverage across all episodes and compressing details into intermediate summaries.

Candidate Generation and Selection. Once the input (original or hierarchical) fits within context, we query the LLM to generate K candidate summaries $\{T_1, T_2, \dots, T_K\}$ via stochastic decoding. This captures variability in abstraction and phrasing. To select a robust final summary, each candidate is embedded into a semantic vector space using a pretrained embedding model, yielding $\phi(T_i) \in \mathbb{R}^d$. We compute the centroid:

$$c = \frac{1}{K} \sum_{i=1}^K \phi(T_i),$$

and measure distances:

$$d_i = \|\phi(T_i) - c\|_2.$$

The final summary T^* is chosen as the *median representative*, i.e. the candidate closest to the median-ranked distance from the centroid:

$$T^* = \arg \min_{T_i} \left| \text{rank}(d_i) - \frac{K}{2} \right|.$$

This selection scheme balances *generality* (summaries near the centroid capture broad regularities) with *specificity* (summaries farther away capture contextual details), yielding a consensus-style narrative.

Algorithm 1 SYSLLM FRAMEWORK

Input: Environment ENV , trained policy π , captioners $C_{\text{obs}}, C_{\text{act}}$, base prompt P

Parameters: number of episodes N , token budget κ , #candidates K , embedding model $\phi(\cdot)$

Output: Policy summary $T^* \in \mathcal{T}$

```

1: Initialize Textual Experience Buffer  $TEB \leftarrow \emptyset$ 
2: for  $i = 1$  to  $N$  do
3:    $t \leftarrow 1$ ;  $o_t \leftarrow ENV.RESET()$ ;  $epReward \leftarrow 0$ ;  $done \leftarrow \text{False}$ 
4:   while  $\neg done$  do
5:      $a_t \sim \pi(\cdot | o_t)$ 
6:      $TEB.ADD(C_{\text{obs}}(o_t), C_{\text{act}}(a_t), epReward, i)$ 
7:      $(o_{t+1}, r_{t+1}, done) \leftarrow ENV.STEP(a_t)$ 
8:      $epReward \leftarrow epReward + r_{t+1}$ ;  $t \leftarrow t + 1$ 
9:      $o_t \leftarrow o_{t+1}$ 
10:  end while
11: end for
12:  $T \leftarrow \text{HIERARCHICALSUMMARIZE}(TEB, \kappa, P, K, \phi)$ 
13: return  $T$ 

```

Algorithm 2 HIERARCHICALSUMMARIZE

Input: textual experience buffer subset D , token budget κ , prompt P , #candidates K , embedding model ϕ

Output: summary $T \in \mathcal{T}$

```

1: if  $\text{TOKENS}(\text{FORMAT}(D)) \leq \kappa$  then
2:    $X \leftarrow P + \text{FORMAT}(D)$ 
3:    $\{T_1, \dots, T_K\} \leftarrow \text{LLM.SAMPLE}(X, K)$ 
4:    $E_i \leftarrow \phi(T_i) \in \mathbb{R}^d \quad \forall i \in \{1, \dots, K\}$ 
5:    $c \leftarrow \frac{1}{K} \sum_{i=1}^K E_i$ 
6:    $d_i \leftarrow \|E_i - c\|_2 \quad \forall i$ ;  $j^* \leftarrow \arg \min_j \left| \text{rank}(d_j) - \frac{K}{2} \right|$ 
7:   return  $T_{j^*}$ 
8: else
9:    $\{D^{(1)}, \dots, D^{(M)}\} \leftarrow \text{PARTITION}(D, \kappa)$   $\triangleright$  disjoint, each fits
10:  for  $m = 1$  to  $M$  do
11:     $S_m \leftarrow \text{HIERARCHICALSUMMARIZE}(D^{(m)}, \kappa, P, K, \phi)$ 
12:  end for
13:  return  $\text{HIERARCHICALSUMMARIZE}(\{S_1, \dots, S_M\}, \kappa, P, K, \phi)$ 
14: end if

```

5 EXPERIMENTAL SETUP

We evaluate SySLLM across controlled reinforcement learning environments that capture both simple and complex agent behaviors. Specifically, we apply our framework to five environments from the MiniGrid suite [10] and the more challenging Crafter environment [18]. These environments were selected to span a range of task structures, observation modalities, and policy complexities. We use the gpt-4-turbo model with a temperature of 0.5 [1], and the text-embedding-3-small model [33] as the embedding function.

5.1 MiniGrid

MiniGrid is a grid-world framework where agents perform goal-directed navigation and object-interaction tasks under partial observability. We instantiate seven agents across five environments, ensuring diversity in both policy performance and behavioral style.

Captioners. To construct the Textual Experience Buffer (TEB), we implement a rule-based captioning system that maps raw grid observations and discrete actions into structured natural-language descriptions. *Observation Captioner* (C_{obs}): generates textual descriptions of visible elements, including object types (e.g., keys, doors, obstacles) and their spatial relationships relative to the agent. *Action Captioner* (C_{act}): translates the discrete action set (e.g., `turn_left`, `move_forward`) into natural-language strings. *Agents and Training* We trained three agents with qualitatively distinct policies in the MiniGrid-Unlock environment:

- **Goal-directed agent:** wide 7×7 observation window, optimized to minimize the steps to unlock the door.
- **Short-sighted agent:** restricted 3×3 observation window, leading to more myopic strategies.
- **Random agent:** selects actions uniformly at random, providing a non-structured behavioral baseline.

In addition, we trained one agent each in four further environments: Dynamic Obstacles, Lava Gap, Red-Blue Doors, and Crossing. All MiniGrid agents were trained using PPO [37, 39] for 1M timesteps per seed for three random seeds. Agent performance statistics are reported in Table 1. The complete hyperparameters are provided in the appendix.

TEB Collection and Summarization. For each agent, we collect 50 evaluation episodes to construct the TEB. Each buffer is formatted into the structured prompt and processed as described in Section 4.3, where candidate summaries are generated via stochastic decoding and post-processed for abstractive summarization. Prompt templates used for the MiniGrid suite domain are listed in the appendix.

5.2 Crafter

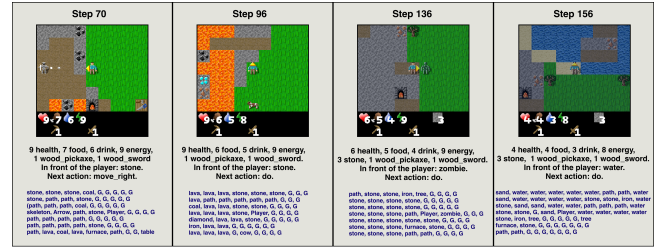
Crafter is a 2D, partially observable world inspired by Minecraft, featuring procedurally generated maps, resource gathering, crafting, and an achievement tree that defines agent progress.

Captioners To adapt the captioning system to Crafter, we extend the MiniGrid captioner to encode: *Observation Captioner*: inventory contents, spatial relations to nearby resources and threats, current health and stamina, and unlocked achievements. *Action Captioner*: maps the Crafter action set into textual forms, e.g., “`move_right`”, “`place_table`”, “`make_wood_sword`”. Figure 3 shows an example trajectory and its textualized representation.

Agents and Training We train two agents with distinct behavior:

- **Resource-Collector agent:** trained with DreamerV3 [19], capable of sustained survival, resource collection, and the crafting of basic tools.
- **Random agent:** uniformly samples from the action set, serving as a baseline with no structured policy.

TEB Collection and Summarization. For each agent, we log 5 evaluation episodes. Due to the long horizon of Crafter, the TEB for each episode can exceed the LLM’s context length. In this case, we apply the hierarchical summarization strategy described in Section 4.3. Prompt templates used for the Crafter domain are listed in the appendix.



Achievements unlocked: collect_coal, collect_drink, collect_sapling, collect_stone, collect_wood, defeat_zombie, make_wood_pickaxe, make_wood_sword, place_furnace, place_plant, place_stone, place_table, wake_up.

Figure 3: Four steps from a trajectory of the Resource-Collector agent in the Crafter environment, alongside their corresponding captions generated using the observation and action captioners. For each step, the captions describe the agent’s inventory status, the object currently in front of it, and the next action selected by the agent. A textual representation of the visible grid (highlighted in blue) is also included to reflect the agent’s local perception. Additionally, all unique achievements unlocked by the agent throughout the trajectory are summarized in red.

Env.	Agent	Mean Reward \pm SD	Mean Length	Success / Achievements
MiniGrid	Unlock Goal-directed	0.73 \pm 0.21	20.25	Success 0.93
	Unlock Short-sighted	0.41 \pm 0.27	44.43	Success 0.77
	Unlock Random	0.00 \pm 0.01	70.00	Success 0.00
	Dynamic Obstacles	0.78 \pm 0.06	17.20	Success 1.00
	Lava Gap	0.90 \pm 0.02	10.82	Success 1.00
	Red Blue Doors	0.70 \pm 0.26	17.06	Success 0.88
Crafter	Resource-Collector	10.43 \pm 2.11	234.6	11.33 achievements
	Random	1.39 \pm 1.19	164.4	2.29 achievements

Table 1: Performance metrics across MiniGrid (500 episodes, 3 seeds) and Crafter (100 episodes, 3 seeds). For MiniGrid, performance is measured by success rate; for Crafter, by the number of unique achievements unlocked.

6 ILLUSTRATIVE POLICY SUMMARIES

To illustrate the summaries produced by SySLLM, we present structured case studies that connect textual motifs to quantitative evidence and then expand them into detailed narrative accounts. This combination shows that the framework produces explanations that are not only linguistically coherent but also grounded in the actual behavior of the agents. A full summary of the Unlock Goal-Directed agent is included in the appendix.

Structured Case Studies. Table 2 reports representative agents in MiniGrid and Crafter. For each, we show: (i) the central claim extracted by SySLLM, and (ii) the quantitative metrics that support or refute this claim. The alignment between narrative motifs and behavioral statistics illustrates that SySLLM captures recurring strategies and limitations in a manner consistent with ground-truth agent performance. For example, the Unlock Goal-Directed agent’s motif of “turning toward the nearest key or door” is supported by a 0.93 success rate and shorter episode lengths, while the Random agent’s “lack of coherent strategy” corresponds to maximal episode lengths and zero success rate. In Crafter, the Resource-Collector agent’s emphasis on survival resources is matched by high reward

and achievement counts, whereas the Random agent’s incoherence is validated by minimal achievement progression.

Expanded Narrative Insights. Beyond single-sentence claims, SySLLM produces multi-faceted descriptions of agent behavior. Figure 4 presents detailed motifs across environments, highlighting how the framework abstracts local decisions into global patterns. These narratives capture not only the strengths of agents (e.g. consistent lava avoidance, structured door-order strategies) but also nuanced inefficiencies (e.g., unnecessary turns in dense obstacle scenarios, repetitive failed crafting attempts in Crafter). Such fine-grained insights demonstrate SySLLM’s ability to diagnose effective heuristics and characteristic failure modes.

Together, these examples demonstrate that abstractive textual summaries can function as faithful and interpretable accounts of reinforcement learning policies.

7 EXPERT EVALUATION OF SUMMARIES

We complement the quantitative performance analysis with an expert-based evaluation of SySLLM summaries. The goal is to assess how well the generated summaries capture the behavioral motifs observed by human experts and to quantify correctness while identifying potential hallucinations.

Evaluation Protocol. We recruited six graduate students with research experience in the training and evaluation of RL agents. The experts were divided into two groups: Experts 1–3 annotated the MiniGrid-Unlock (goal-directed and short-sighted) agents, while Experts 4–6 annotated the remaining MiniGrid agents. In addition, Experts 1, 2, and 6 annotated the Crafter agents. Each expert was shown a 120-second video per agent, depicting representative trajectories. Based on these trajectories, the experts were instructed to produce textual summaries using the same SySLLM prompting guidelines (see the appendix). This alignment ensures comparability between expert- and model-generated summaries.

Scoring Framework. To compare SySLLM summaries S_{LLM} with expert summaries $\{S_{\text{exp}}^j\}_{j=1}^m$, we decompose both into sets of atomic key points, denoted \mathcal{K}_{LLM} and $\mathcal{K}_{\text{exp}}^j$, respectively. Semantic equivalence is assessed at the level of atomic propositions, where each key point expresses a single behavioral claim. Following standard practice in evaluating human-interpretable explanations [21], propositions are manually extracted and matched without embedding-based or heuristic similarity measures. Each SySLLM key point is labeled as a full match, partial match, or unsupported relative to the expert set. This procedure yields a transparent and reproducible matching process, with substantial inter-annotator agreement (AC1 = 0.72). The full annotation protocol is provided in the appendix.

Each pairwise comparison is scored as:

$$\text{match}(k_{\text{LLM}}, k_{\text{exp}}) = \begin{cases} 1 & \text{if semantically equivalent,} \\ 0.5 & \text{if partially overlapping,} \\ 0 & \text{otherwise.} \end{cases}$$

Recall. Recall measures the extent to which the model summary covers expert-identified key points:

$$\text{Recall} = \frac{1}{m} \sum_{j=1}^m \frac{\sum_{k \in \mathcal{K}_{\text{exp}}^j} \max_{k' \in \mathcal{K}_{\text{LLM}}} \text{match}(k, k')}{|\mathcal{K}_{\text{exp}}^j|}.$$

Precision. Precision measures the correctness of SySLLM key points relative to expert judgments. For each expert j , annotators are shown the set difference $\mathcal{K}_{\text{LLM}} \setminus \mathcal{K}_{\text{exp}}^j$ and asked to label each key point as Matched, Partially Matched, or Not Matched. Precision is defined as:

$$\text{Precision} = \frac{\sum_{k \in \mathcal{K}_{\text{LLM}}} \max_j \text{match}(k, \mathcal{K}_{\text{exp}}^j)}{|\mathcal{K}_{\text{LLM}}|}.$$

This formulation captures both coverage (recall) and correctness (precision), while enabling explicit identification of hallucinated content through unmatched model key points.

Results. Table 3 reports per-agent recall and precision. Recall scores range from 0.687 (Unlock goal-directed) to 0.914 (Crossing), with a mean of 0.840, demonstrating substantial overlap with expert-identified points. Precision scores range from 0.769 (Dynamic Obstacles) to 0.864 (Short-Sighted), with a mean of 0.839, indicating minimal hallucination. For example, in the Crossing environment, the point “*The agent frequently checks for walls in its path and adjacent tiles*” was rejected by all experts, illustrating a rare hallucination.

Inter-Annotator Agreement. To ensure the reliability of expert annotations, we calculated both raw agreement (percentage of identical match scores across experts) and Gwet’s AC1 coefficient, which is robust to class imbalance in categorical judgments. Across all agents, the mean raw agreement was 70%, while the mean AC1 reached 0.72, indicating substantial inter-rater reliability. This ensures that the observed recall/precision metrics are not artifacts of inconsistent annotations.

Taken together, these results show that SySLLM summaries exhibit both high coverage and correctness relative to expert annotations, with recall and precision consistently above 0.8.

8 USER STUDY

We conducted a controlled user study to evaluate the usefulness of SySLLM summaries compared to HIGHLIGHTS-DIV (HIGHLIGHTS) [3], a standard demonstration-based benchmark in XRL which selects a set of high importance and diverse execution trajectories. The study assessed both subjective preferences and objective task performance. We focus on three qualitatively distinct agents from the MiniGrid Unlock environment (goal-directed, short-sighted, and random), ensuring the diversity of policies from structured strategies to noisy behaviors. For completeness, we provide a short description of HIGHLIGHTS in the appendix.

Experimental Design. The study used a mixed design with two tasks. Task 1 (Preferences) followed a within-subject setup: each participant evaluated both modalities (SySLLM and HIGHLIGHTS), with order counterbalanced to mitigate ordering effects. Task 2 (Identification) followed a between-subject setup: Participants viewed only one modality, aligned with their Task 1 order. This produced four experimental conditions that varied by summary modality and agent type, as shown in Table 4. For HIGHLIGHTS, highlight videos were generated using 300 traces, a context length of 5, and 20 highlights, while SySLLM summaries were generated from 50 captioned episodes (see Section 5).

Procedure. The participants first completed a tutorial on MiniGrid Unlock rules, followed by a comprehension quiz. In Task 1,

Env.	Agent	SySLLM Summary Claim	Quantitative Alignment
MiniGrid	Unlock Goal-directed	Turns toward nearest key/door	Success 0.93, mean length 20.25
	Unlock Random	No clear strategy	Success 0.00, max length episodes
	Dynamic Obstacles	Avoids obstacles, occasional inefficiency	Success 1.00, higher variance in length
	Lava Gap	Jumps gap consistently	Success 1.00, mean length 10.82
Crafter	Resource-Collector	Focused, survival-oriented strategy; Unlocks early achievements	Reward 10.43, 11.33 unique achievements
	Random	Sporadic, incoherent progression	Reward 1.39, 2.29 unique achievements

Table 2: Structured case studies of SySLLM summaries across environments. Extracted motifs are validated against quantitative metrics, showing alignment between narrative claims and behavioral statistics.

<p>Unlock – Goal-directed</p> <ol style="list-style-type: none"> The agent effectively identifies keys and adjusts its path based on their relative position, shifting focus to unlocking the door. It consistently turns towards the nearest key or door, minimizing distance, which remains consistent across episodes. The agent completes episodes efficiently, averaging 15–25 steps with near-maximal cumulative rewards. 	<p>Crossing</p> <ol style="list-style-type: none"> The agent moves towards the green goal once it enters its field of vision, adjusting its path accordingly. It avoids collisions with walls through timely directional changes. 	<ol style="list-style-type: none"> Predictable Behavior: Episodes are characterized by high consistency in resource collection and basic tool crafting actions, with low variance across episodes. Inconsistent Health and Exploration Management: While the agent effectively manages drink levels, it shows less consistency in food and health management and sacrifices exploration efficiency for achievement unlocking.
<p>Unlock – Short-sighted</p> <ol style="list-style-type: none"> The agent follows a right-wall method, moving forward until encountering an obstacle before turning. It identifies keys and doors efficiently, maneuvering toward and using them correctly. Decisions are heavily influenced by its immediate field of vision, reacting only to nearby objects. 	<p>Dynamic Obstacles</p> <ol style="list-style-type: none"> The agent effectively avoids moving obstacles (blue balls) by adjusting its movement. It identifies objects in its field of vision and makes informed navigation decisions. In dense obstacle scenarios, occasional inefficiencies or unnecessary turns are observed. 	<p>Crafter – Random</p> <ol style="list-style-type: none"> Ineffective Crafting and Resource Management: The agent frequently attempts crafting without the necessary resources or understanding of prerequisites, leading to repeated failures and minimal progress in achieving complex objectives. Poor Survival Strategy: The agent consistently demonstrates ineffective survival behavior, including health depletion and poor management of food and drink levels, which hampers its ability to sustain itself in the game environment. Limited Achievement Progression: While the agent reliably unlocks basic achievements like <code>wake_up</code> and <code>collect_sapling</code>, it struggles to achieve more complex milestones that require crafting, resource management, or combat engagement. Repetitive and Ineffective Actions: Episodes are marked by high frequencies of ‘noop’ actions and repetitive failed attempts at crafting, reflecting a lack of strategic adaptation and learning from past failures. Lack of Combat Engagement: The agent shows minimal engagement with combat mechanics and fails to defend effectively against threats such as zombies and skeletons. Predictable Behavior: Across episodes, the agent exhibits consistent, ineffective patterns of action, suggesting significant limitations in its decision-making processes and adaptability.
<p>Unlock – Random</p> <ol style="list-style-type: none"> The agent exhibits unstructured behavior, often repeating unnecessary actions. It frequently toggles doors multiple times or picks up and drops keys without using them effectively. 	<p>Crafter – Resource-collector</p> <ol style="list-style-type: none"> Strong Focus on Resource Collection: The agent consistently prioritizes gathering essential resources such as wood, stone, and drink, which are foundational for crafting tools and maintaining basic survival metrics. Effective Basic Tool Crafting: Regular crafting of basic tools like wood pickaxes and swords enables the agent to enhance resource collection and engage in occasional combat. Achievement Unlocking: The agent reliably unlocks achievements related to resource collection and basic tool crafting but struggles with more advanced achievements, highlighting a potential area for improvement. Moderate Combat Engagement: The agent occasionally engages with zombies, using crafted tools for defense, showing moderate adaptability to threats but limited combat readiness overall. 	
<p>Lava Gap</p> <ol style="list-style-type: none"> The agent consistently avoids lava, demonstrating awareness of environmental hazards. Upon encountering an obstacle, it either turns or moves in the opposite direction. 		
<p>Red-Blue</p> <ol style="list-style-type: none"> The agent prioritizes opening the red door before the blue door, optimizing reward accumulation. It successfully interacts with doors in a structured sequence, adhering to task constraints. 		

Figure 4: Insights from agents’ SySLLM summaries in the MiniGrid environments.

they watched a 120 seconds video of an agent’s behavior and then rated a summary (SySLLM or HIGHLIGHTS) on eight explanation quality metrics (7-point Likert), adapted from Hoffman et al. [21]. After evaluating both modalities, they provided direct preference judgments: which summary better reflected the agent’s policy and by what margin. In Task 2, participants were shown a summary (textual or visual) and asked to match it to one of three short (20s) videos: the correct agent plus two distractors. Each participant completed three trials (Q1: goal-directed, Q2: random, Q3: short-sighted). For each, they indicated their choice, rated confidence, and provided a justification.

Participants. We recruited 200 participants from Prolific (native English speakers from the US, UK, Canada, and Australia). Compensation was £3.75 base plus a £1 bonus for correct completion of Task 2. After exclusions for failed attention checks and implausibly short completion times (below 300 seconds), 192 participants remained (94 female, $M_{age} = 36.4$, $SD = 12.1$).

Results. In Task 1, SySLLM consistently outperformed HIGHLIGHTS across all metrics (Fig. 5). Paired t -tests confirmed the

difference as highly significant ($T = 13.99$, $p < 10^{-33}$). Direct preference questions reinforced this: 75.5% of participants favored SySLLM, and the comparative Likert rating averaged $M = 5.97$, $SD = 1.44$ (neutral baseline = 4). Qualitative feedback highlighted that SySLLM explained why agents acted as they did, while HIGHLIGHTS required subjective inference. For example: “There are instances in the video where the agent seems to turn random corners. The summary explains why.” In Task 2, correctness rates for both modalities exceeded random-guess baselines (Fig. 6). Chi-Square tests found no significant differences between SySLLM and HIGHLIGHTS across Q1–Q3. However, confidence scores revealed a significant effect in Q3, where participants in the SySLLM condition reported higher confidence ($t = 3.42$, $p = 0.0008$).

Overall, participants rated SySLLM summaries as significantly clearer and more informative than the highlight videos. Although both modalities supported correct agent identification, textual summaries provided stronger interpretive cues, particularly reflected in higher confidence for certain agents. These results suggest that abstractive, language-based policy summaries enhance subjective

Agent	Expert	Recall	Precision	Mean
Unlock Goal-Directed	E1	0.500	0.864	$R = 0.687, P = 0.864$
	E2	0.643	0.864	
	E3	0.917	0.864	
Unlock Short-Sighted	E1	0.800	0.846	$R = 0.878, P = 0.839$
	E2	0.833	0.807	
	E3	1.000	0.923	
Dynamic Obstacles	E4	0.583	0.692	$R = 0.739, P = 0.769$
	E5	0.833	0.692	
	E6	0.800	0.923	
Lava Gap	E4	0.667	0.769	$R = 0.794, P = 0.811$
	E5	0.786	0.846	
	E6	0.929	0.818	
Red-Blue Doors	E4	0.857	0.767	$R = 0.871, P = 0.834$
	E5	0.857	0.867	
	E6	0.900	0.867	
Crossing	E4	0.750	0.731	$R = 0.914, P = 0.795$
	E5	0.917	0.808	
	E6	1.000	0.846	
Crafter Resource-Collector	E7	0.938	0.893	$R = 0.931, P = 0.871$
	E8	0.938	0.857	
	E9	0.917	0.864	
Crafter Random	E7	0.929	0.917	$R = 0.902, P = 0.929$
	E8	1.000	0.958	
	E9	0.778	0.857	

Table 3: Per-expert recall and precision scores for SySLLM summaries, with aggregated per-agent means. R = Recall, P = Precision. Overall averages across all agents: $R = 0.840, P = 0.839$.

Condition	Task 1 Sequence	Task 1 Agent Type	Task 2 Summary Type
1	SySLLM → HIGHLIGHTS	Goal-directed Agent	SySLLM
2	SySLLM → HIGHLIGHTS	Short-sighted Agent	SySLLM
3	HIGHLIGHTS → SySLLM	Goal-directed Agent	HIGHLIGHTS
4	HIGHLIGHTS → SySLLM	Short-sighted Agent	HIGHLIGHTS

Table 4: Experimental conditions.

interpretability while maintaining competitive performance in behavior recognition.

9 DISCUSSION AND FUTURE WORK

We introduced SySLLM, a framework for *abstractive textual* explanation of reinforcement learning policies. By converting state and action trajectories into structured language and leveraging large language models to generate policy summaries, SySLLM abstracts low-level decision traces into high-level behavioral patterns. Empirical results show close alignment with expert interpretations and a clear user preference over visual demonstrations, supporting language-based abstraction for interpreting complex policies.

Several limitations remain. SySLLM currently relies on domain-specific captioning functions to translate trajectories into text. While feasible in simulated environments, extending to high dimensional or partially observed domains will require perceptual grounding via visual-language pipelines or pretrained vision-language models capable of zero-shot scene understanding.

Future work extends beyond static summarization. One direction is *interactive policy querying*, where users ask natural language questions about agent behavior, enabling counterfactual and

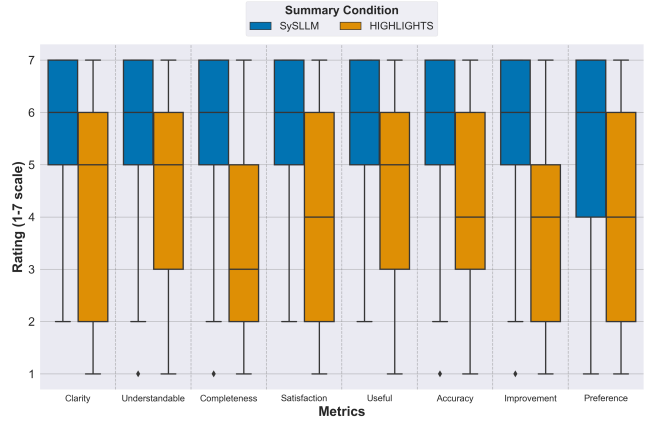


Figure 5: Participant ratings for Task 1 on a 1–7 Likert scale. SySLLM ratings are significantly higher than HIGHLIGHTS ratings across all metrics.

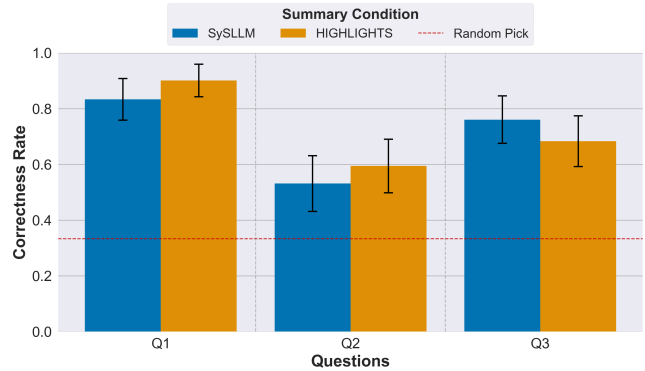


Figure 6: Correctness rate in Task 2. Error bars indicate 95% confidence intervals. Both SySLLM and HIGHLIGHTS outperform the random guess baseline. The differences in correctness rate between SySLLM and HIGHLIGHTS is not statistically significant.

rationale-based explanations. Another is *comparative summarization*, contrasting policies across training stages, reward functions, or architectures. Finally, integrating textual and visual modalities and advancing toward *open-domain policy summarization* may enable benchmark-agnostic systems that characterize agent behavior at scale.

ACKNOWLEDGMENTS

Funded by the European Union (ERC, Convey, 101078158) and the Israel Science Foundation (ISF) under Grant No. 3079/24. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

REFERENCES

- [1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).
- [2] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691* (2022).
- [3] Dan Amir and Ofra Amir. 2018. Highlights: Summarizing agent behavior to people. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. 1168–1176.
- [4] Ofra Amir, Finale Doshi-Velez, and David Sarne. 2018. Agent strategy summarization. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. 1203–1207.
- [5] Yotam Amitai and Ofra Amir. 2023. A Survey of Global Explanations in Reinforcement Learning. In *Explainable Agency in Artificial Intelligence*. CRC Press, 21–42.
- [6] Andrew Anderson, Jonathan Dodge, Amrita Sadarangani, Zoe Juozapaitis, Evan Newman, Jed Irvine, Souti Chattopadhyay, Matthew Olson, Alan Fern, and Margaret Burnett. 2020. Mental models of mere mortals with explanations of reinforcement learning. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 10, 2 (2020), 1–37.
- [7] Akanksha Atrey, Kaleigh Clary, and David Jensen. 2019. Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning. *arXiv preprint arXiv:1912.05743* (2019).
- [8] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. 2018. Verifiable reinforcement learning via policy extraction. *Advances in neural information processing systems* 31 (2018).
- [9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.
- [10] Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. 2021. Minimalistic gridworld environment for openai gym (2018). URL <https://github.com/maximecb/gym-minigrid> 6 (2021).
- [11] Youri Coppens, Kyriakos Efthymiadis, Tom Lenaerts, Ann Nowé, Tim Miller, Rosina Weber, and Daniele Magazzeni. 2019. Distilling deep reinforcement learning policies in soft decision trees. In *Proceedings of the IJCAI 2019 workshop on explainable artificial intelligence*. 1–6.
- [12] Shripad Vilasrao Deshmukh, Arpan Dasgupta, Balaji Krishnamurthy, Nan Jiang, Chirag Agarwal, Georgios Theodorou, and Jayakumar Subramanian. 2023. Explaining rl decisions with trajectories. *arXiv preprint arXiv:2305.04073* (2023).
- [13] Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. 2023. Guiding pretraining in reinforcement learning with large language models. In *International Conference on Machine Learning*. PMLR, 8657–8677.
- [14] Upol Ehsan, Brent Harrison, Larry Chan, and Mark O Riedl. 2018. Rationalization: A neural machine translation approach to generating natural language explanations. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*. 81–87.
- [15] Julius Frost, Olivia Watkins, Eric Weiner, Pieter Abbeel, Trevor Darrell, Bryan Plummer, and Kate Saenko. 2022. Explaining reinforcement learning policies through counterfactual trajectories. *arXiv preprint arXiv:2201.12462* (2022).
- [16] Samuel Greycanus, Anurag Koul, Jonathan Dodge, and Alan Fern. 2018. Visualizing and understanding atari agents. In *International conference on machine learning*. PMLR, 1792–1801.
- [17] Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. 2023. Leveraging Pre-trained Large Language Models to Construct and Utilize World Models for Model-based Task Planning. *arXiv:2305.14909 [cs.AI]*
- [18] Danijar Hafner. 2021. Benchmarking the Spectrum of Agent Capabilities. *arXiv preprint arXiv:2109.06780* (2021).
- [19] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. 2023. Mastering Diverse Domains through World Models. *arXiv preprint arXiv:2301.04104* (2023).
- [20] Bradley Hayes and Julie A Shah. 2017. Improving robot controller transparency through autonomous policy explanation. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 303–312.
- [21] Robert R Hoffman, Shane T Mueller, Gary Klein, and Jordan Litman. 2018. Metrics for explainable AI: Challenges and prospects. *arXiv preprint arXiv:1812.04608* (2018).
- [22] Tobias Huber, Benedikt Limmer, and Elisabeth André. 2022. Benchmarking perturbation-based saliency maps for explaining atari agents. *Frontiers in Artificial Intelligence* 5 (2022), 903875.
- [23] Yiding Jiang, Shixiang Shane Gu, Kevin P Murphy, and Chelsea Finn. 2019. Language as an abstraction for hierarchical deep reinforcement learning. *Advances in Neural Information Processing Systems* 32 (2019).
- [24] Zoe Juozapaitis, Anurag Koul, Alan Fern, Martin Erwig, and Finale Doshi-Velez. 2019. Explainable reinforcement learning via reward decomposition. In *IJCAI/ECAL Workshop on explainable artificial intelligence*.
- [25] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems* 35 (2022), 22199–22213.
- [26] Fangjun Li, David C Hogg, and Anthony G Cohn. 2024. Advancing spatial reasoning in large language models: An in-depth evaluation and enhancement using the stepgame benchmark. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 18500–18507.
- [27] Haozhe Liu, Mingchen Zhuge, Bing Li, Yuhui Wang, Francesco Faccio, Bernard Ghanem, and Jürgen Schmidhuber. 2023. Learning to identify critical states for reinforcement learning from videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1955–1965.
- [28] Prashan Madumal, Tim Miller, Liz Sonenberg, and Frank Vetere. 2020. Explainable reinforcement learning through a causal lens. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 2493–2500.
- [29] Joe McCalmon, Thai Le, Sarra Alqahtani, and Dongwon Lee. 2022. Caps: Comprehensive abstract policy summaries for explaining reinforcement learning agents. In *nt’l Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*.
- [30] James McCarthy, Rahul Nair, Elizabeth Daly, Radu Marinescu, and Ivana Dusparic. 2022. Boolean Decision Rules for Reinforcement Learning Policy Summarisation. *arXiv preprint arXiv:2207.08651* (2022).
- [31] Stephanie Milani, Nicholay Topin, Manuela Veloso, and Fei Fang. 2022. A Survey of Explainable Reinforcement Learning. *arXiv preprint arXiv:2202.08434* (2022).
- [32] Suvir Mirchandani, Siddharth Karamcheti, and Dorsa Sadigh. 2021. Ella: Exploration through learned language abstraction. *Advances in neural information processing systems* 34 (2021), 29529–29540.
- [33] OpenAI. 2024. Text Embedding Models: text-embedding-3-small and text-embedding-3-large. <https://platform.openai.com/docs/guides/embeddings>.
- [34] Zahra Parham, Vi Tching de Lille, and Quentin Cappart. 2023. Explaining the Behavior of Reinforcement Learning Agents Using Association Rules. In *International Conference on Learning and Intelligent Optimization*. Springer, 107–120.
- [35] Xiangyu Peng, Mark Riedl, and Prithviraj Ammanabrolu. 2022. Inherently explainable reinforcement learning in natural language. *Advances in Neural Information Processing Systems* 35 (2022), 16178–16190.
- [36] Nikaash Puri, Sukriti Verma, Piyush Gupta, Dhruv Kayastha, Shripad Deshmukh, Balaji Krishnamurthy, and Sameer Singh. 2019. Explain your move: Understanding agent actions using specific and relevant feature attribution. *arXiv preprint arXiv:1912.12191* (2019).
- [37] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research* 22, 268 (2021), 1–8. <http://jmlr.org/papers/v22/20-1364.html>
- [38] Amir Samadi, Konstantinos Koufos, Kurt Debattista, and Mehrdad Dianati. 2024. SAFE-RL: Saliency-aware counterfactual explainer for deep reinforcement learning policies. *IEEE Robotics and Automation Letters* (2024).
- [39] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).
- [40] Pedro Sequeira and Melinda Gervasio. 2020. Interestingness elements for explainable reinforcement learning: Understanding agents’ capabilities and limitations. *Artificial Intelligence* 288 (2020), 103367.
- [41] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291* (2023).
- [42] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems* 35 (2022), 24824–24837.
- [43] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.
- [44] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* 1, 2 (2023).