WALL-EVE: World Alignment with Rule Learning in Visual Environments

Siyu Zhou[1], Tianyi Zhou[2(✉)], Yijun Yang[3], Guodong Long[1(✉)], Deheng Ye[4], Jing Jiang[1], and Chengqi Zhang[5]

[1] AAII, University of Technology Sydney, Sydney, Australia
[2] Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE
[3] Tsinghua University, Beijing, China [4] National University of Singapore, Singapore
[5] The Hong Kong Polytechnic University, Hong Kong SAR, China
Siyu.Zhou-2@student.uts.edu.au, Tianyi.Zhou@mbzuai.ac.ae,
yijun.steven.yang@gmail.com, Guodong.Long@uts.edu.au

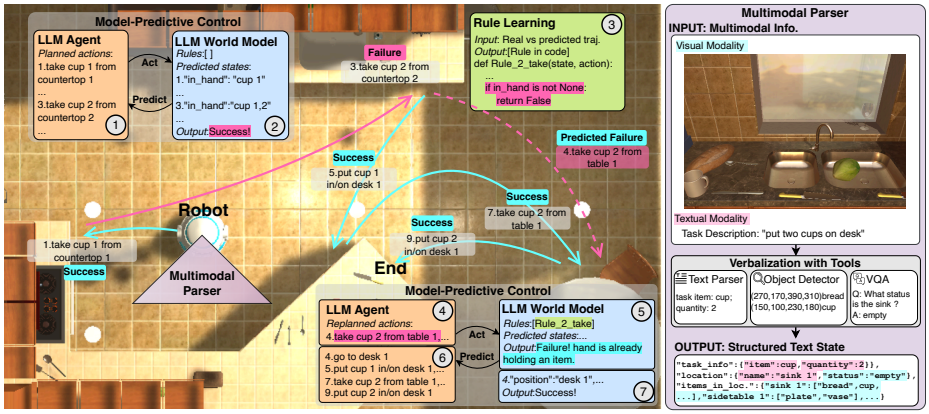

Fig. 1: Illustration of our WALL-EVE agent putting two cups on the same table in ALFWorld. (Left) Step 1-2: the agent makes a decision via MPC with the initial unaligned world model, resulting in a failed action for taking cup 2. Step 3: by comparing real trajectories with the world model predictions, WALL-EVE learns a critical rule that if the agent's hand is already holding an item, the action of taking another item will fail. Step 4-5: the learned rule helps the world model make accurate predictions for transitions that were predicted mistakenly in MPC. Step 6-7: the agent adjusts its decision accordingly, placing cup 1 on the table before picking up cup 2 to continue completing the task. (Right) Overview of the multimodal parser, which integrates textual and visual inputs into a unified JSON-formatted state using a text parser, object detector, and visual question answering (VQA) module.

Abstract. Constructing world models for AI agents in visual environments has long been a significant challenge, chiefly due to the difficulty

✉ Corresponding authors.

of achieving precise alignment of the dynamics between the world model and the environment, given noisy pixel-level observations. Existing world models solely rely on video or image generation, which are exceedingly complex and expensive to train, yet still suffer from hallucinations and violations of basic rules due to misalignment. In this paper, we show that a few rules suffice to convert a pretrained large language model (LLM) to an accurate world model for a specific visual environment. We propose a training-free approach to efficiently learn these complementary rules from multimodal agent trajectories collected in the visual environment. In particular, we convert the multimodal input into structured text and extract symbolic rules using an LLM by comparing agents' true trajectories with world model predictions if they have conflicts. We use this neurosymbolic approach to induce, update, and prune rules during exploration, resulting in a precise world model. It allows us to build a novel model-predictive control (MPC) agent, "WALL-EVE", without training any policy. Before taking each action, WALL-EVE utilizes an LLM as a look-ahead optimizer to determine the next k-step actions through interactions with our world model. On challenging tasks in ALFWorld and Minecraft, WALL-EVE achieves higher success rates than existing methods, while reducing inference times and the number of tokens required for reasoning. In ALFWorld, WALL-EVE surpasses the state-of-the-art method RAFA with only **17%** of RAFA's token usage. In Minecraft, WALL-EVE outperforms baselines by **8-30%** in success rate but costs **8**–**20** fewer replanning rounds and only **60**–**85%** tokens. Code is available here.

Keywords: Embodied agent · Large language Model · World model

1 Introduction

Constructing accurate and efficient world models for visual embodied environments is critical to AI-agent development but remains a formidable challenge. These models are crucial for simulating, predicting, and planning future trajectories, given past interactions and intended actions [3, 44]. They enable agents to understand their surroundings, make informed decisions, and execute tasks with precision. However, achieving accurate alignment between the world model and the specific dynamics of the deployed environment is a significant obstacle that hinders the effectiveness of world model-based agents.

Current mainstream approaches to building world models for visual environments predominantly rely on video generation. These models aim to directly synthesize realistic videos for future states, leveraging advances in video generation technologies [1, 20, 32]. Despite their potential, these approaches mainly focus on low-level details but lack semantic-level alignment with environmental dynamics, resulting in violations of basic rules and hallucinations. Specifically, ensuring that generated videos or images accurately reflect the underlying physical and temporal properties of the environment is inherently difficult [24]. Unlike language, which benefits from well-established structures and controllable generation processes, video and image generation are less constrained and more

susceptible to discrepancies between generated content and real-world dynamics [7,43,63].

Establishing alignment for video or image generation-based world models typically involves extensive training to fine-tune them to accurately reflect the environment's dynamics, a process that is both time-consuming and resource-intensive. The high demand for training resources often renders these approaches prohibitively expensive, limiting their scalability and applicability across diverse and complex environments [36,43,46].

Addressing these challenges necessitates a paradigm shift in how world models are constructed for visual embodied environments. In this paper, we introduce an efficient, training-free method for building a world model that circumvents the extensive resource demands of video and image generation approaches. Our method leverages the strengths of Large Language Models (LLMs) by verbalizing visual inputs to transform them into a structured text-based representation of the visual world. This transformation allows us to exploit the well-controlled and structured nature of language to achieve alignment between the LLM-based world model and the environment's dynamics. By tapping into the rich prior knowledge embedded within LLMs, we streamline the alignment process, making it more resource-efficient and scalable.

To achieve alignment between the LLM world model and the environment's dynamics, we propose training-free techniques centered around rule learning in our framework called "World Alignment with Rule Learning in Visual Environments (WALL-EVE)". WALL-EVE builds a neurosymbolic world model by learning complementary rules using LLMs' inductive reasoning and code generation capabilities. To be specific, WALL-EVE interacts with the environment to collect real trajectories, verbalizes the visual inputs, and compares these text-based representations with the predictions of the LLM world model. The comparison results are then analyzed by an LLM, which extracts new rules or modifies existing ones to improve the consistency between the predicted and real trajectories. To keep the rule set minimal and necessary, we prune the rules by solving a maximum coverage problem, selecting a subset of rules that maximally cover the transitions the LLM alone failed to predict. This iterative rule learning procedure continues until the combination of the LLM and

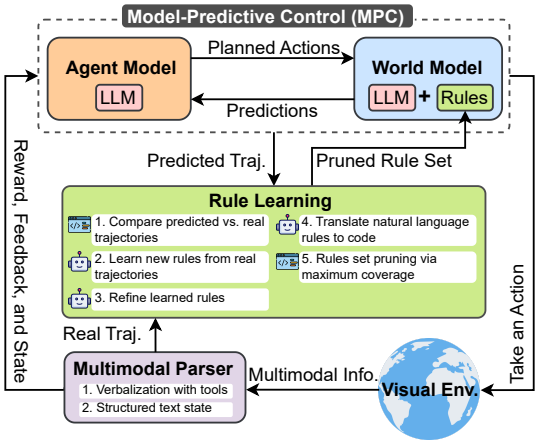

Fig. 2: Overview of WALL-EVE. The agent's action per step is controlled by MPC, where the agent model plans actions in a look-ahead window based on the LLM+rules world model's predictions.

the learned rules performs as an accurate world model. Importantly, this alignment process does not rely on gradient-based training, significantly reducing the computational overhead traditionally associated with world model development.

The precise neurosymbolic world model achieved by WALL-EVE enables stronger model-based LLM agents for challenging open-world tasks. However, model-based reinforcement learning (RL) of LLM agents in complex visual environments is still hindered by the expensive exploration and fine-tuning of LLMs. We therefore revisit the classical idea of model-predictive control (MPC) [17–19,37], which, unlike RL, does not require training a policy network but optimizes actions over a look-ahead time window at each step. To reduce per-step optimization cost, we use the LLM agent as an optimizer, searching for the optimal look-ahead actions by interacting with the LLM world model established via visual verbalization and rule alignment in WALL-EVE. With an aligned LLM world model and an efficient LLM-based optimizer, MPC leads to a more promising and efficient framework for LLM agents in open-world environments.

We evaluate WALL-EVE on challenging open-world environments such as ALFWorld and Minecraft, in which agents need to explore large-scale landscapes, understand multi-modal inputs, and target complicated tasks. **Our contributions are threefold**: (i) We investigate the underexplored challenge of world alignment for LLM agents. (ii) We propose a novel class of neurosymbolic world models based on rule learning and visual verbalization with LLMs. (iii) We develop an LLM agent excelling in visual environments on top of MPC using the aligned world model.

2 Related Work

Recent world models for visual environments are resource-intensive due to their reliance on video or image generation [1,20,32,43,52]. Our approach circumvents these demands by employing visual verbalization to construct a parallel text representation, enabling the usage of the LLM-based world model in a training-free manner. Unlike previous work aligning LLMs with environments through fine-tuning or using hand-crafted prompts [5,6,11,12,14,29,53,54], our method achieves automatic alignment via rule learning from the agent-explored trajectories, reducing human intervention and enhancing performance. While integrating LLMs with rule learning has improved reasoning and generalization in various tasks [31, 55, 57, 60, 62], prior work is always model-free and has not focused on aligning LLM-based world models with environment dynamics. Our research fills this gap and first develops an MPC-based agentic framework. Please refer to Appendix A for a more comprehensive discussion.

3 Method

3.1 Multimodal Parser

Consider an LLM agent π_{LLM} interacting with a visual environment over a fixed horizon length H. At each time step h, the agent observes the environment state

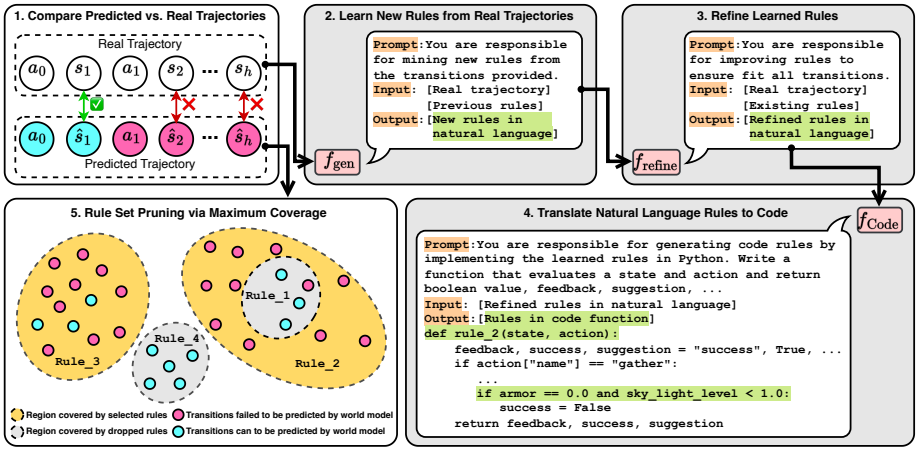

Fig. 3: Rule learning details. Rule learning iteratively refines the rules by comparing the world model's predictions with the agent's actual trajectories in the environment, which takes five steps: (**1**) comparing predicted and actual trajectories; (**2**) learning new rules from real trajectories; (**3**) refining learned rules on all previous transitions; (**4**) translating text rules to code; and (**5**) rule set pruning via solving a maximum coverage problem in Eq. 3. (**2**)-(**4**) are handled by LLMs, while (**1**) and (**5**) are executed by programs.

$s_h = (s_h^t, s_h^v)$, comprising textual information s_h^t (e.g., task descriptions) and visual input s_h^v (e.g., agent's first-person view). To effectively leverage these observations, we introduce a multimodal parser f_{parser}, converting s_h into a structured JSON representation: $s_h^J = f_{\text{parser}}(s_h^t, s_h^v)$, as shown in Figure 1. Specifically, s_h^t is structured into standardized text, while s_h^v is processed through an object detector and a visual question answering (VQA) module to extract relevant objects and context. This unified representation s_h^J replaces the raw state s_h, facilitating structured learning of LLM-based world models. Using s_h^J, the LLM agent selects an action $a_h = \pi_{\text{LLM}}(s_h^J)$ and observes the subsequent state s_{h+1}^J. Each transition is denoted by $\delta_h = (s_h^J, a_h, s_{h+1}^J)$, and an episode trajectory is represented as $\tau = (s_0^J, a_0, s_1^J, \ldots, s_{H-1}^J, a_{H-1}, s_H^J)$. For brevity, we hereafter denote s^J simply as s.

3.2 Model-Predictive Control (MPC) Agent

We integrate Model-Predictive Control (MPC) with our LLM-based world model f_{wm} to enhance agent planning and decision-making (Figure 2). Given a current state-action pair (s_h, a_h), the world model predicts the next state as $\hat{s}_{h+1} = f_{\text{wm}}(s_h, a_h)$. MPC aims to determine the optimal action sequence $A^* = \{a_{h+i}\}_{i=0}^{I-1}$ that maximizes the expected cumulative reward over a finite horizon H:

$$A^* = \arg\max_A \sum_{i=0}^{H-1} \gamma^i \mathcal{F}(\hat{s}_{h+i+1}), \tag{1}$$

where $\gamma \in [0, 1]$ is the discount factor and $\mathcal{F}(\cdot)$ is the reward function evaluated on the predicted next state. Misalignment between predicted and true dynamics may lead to inaccurate rewards and suboptimal or unsafe actions. Addressing this misalignment is essential for reliable MPC performance.

3.3 World Alignment with Rule Learning in Visual Environment (WALL-EVE)

Accurately predicting next states in open-world visual environments is challenging due to complex and variable dynamics. To mitigate misalignment between the LLM world model f_{wm} and the true environment, we propose a rule learning framework (Figure 3) that leverages LLMs' inductive reasoning and code generation capabilities. By extracting symbolic rules from structured multimodal inputs, this framework aligns f_{wm} with the real environment, improving prediction accuracy and downstream MPC performance.

Comparing Predicted and Real Trajectories. To detect misalignments, we compare the predicted next state $\hat{s}_{h+1}$ with the actual s_{h+1}. Specifically, our world model first predicts whether a transition will succeed or fail, represented by a binary variable $\hat{suc} \in \{0, 1\}$, and then predicts the next state s_{h+1}, since s_{h+1} can be directly derived from the success/failure, e.g., if "craft a table" succeeds, the number of tables would increase by one and the consumed materials would decrease in s_{h+1}. This prediction process resembles chain-of-thought [50], and can be defined as:

$$\hat{suc}_h = f_{\text{wm}}(s_h, a_h),\ \hat{s}_{h+1} = f_{\text{wm}}(s_h, a_h, \hat{suc}_h). \tag{2}$$

By contrast, directly predicting s_{h+1} and comparing the prediction with the actual state can be overly complex and prone to noises from random factors (e.g., weather or creature presence). Thus, we only compare the predicted vs. actual transition success/failure.

We classify transitions into correct ($\delta^{\text{cor}} \in \mathcal{D}^{\text{cor}}$) and incorrect ($\delta^{\text{inc}} \in \mathcal{D}^{\text{inc}}$) predictions, as shown in Step 1 of Figure 3. Then, real trajectories will be used to induce new rules, and $\mathcal{D}^{\text{inc}}$ can be adopted to prune redundant ones, aligning the world model with real dynamics efficiently.

Learning New Rules from Real Trajectories. Before addressing these misalignments, we prompt the LLM f_{gen} to generate new natural language rules from environmental trajectories τ^{real} (see Appendix B.1 for detailed prompt). f_{gen} produces new natural language rules $R^{\text{NL}}_{\text{new}}$ that explain the observed dynamics, ensuring they are distinct from previous rules $R^{\text{NL}}_{\text{prev}}$: $R^{\text{NL}}_{\text{new}} = f_{\text{gen}}(\tau^{\text{real}}, R^{\text{NL}}_{\text{prev}})$.

Refining Learned Rules. Then, we prompt the LLM to update existing rules based on the real trajectories τ^{real} (see Appendix B.2 for detailed prompts). Rules learned in the early exploration stage could be inaccurate due to the limited data coverage, so the LLM needs to identify conflicting rules and update

them as needed. The set of all existing rules is denoted to $R^{\mathrm{NL}} = R^{\mathrm{NL}}_{\mathrm{prev}} \cup R^{\mathrm{NL}}_{\mathrm{new}}$, where the LLM f_{refine} refines these rules with the real trajectories: $R^{\mathrm{NL}} = \{r^{\mathrm{NL}}_i\}_{i=1}^{N} = f_{\mathrm{refine}}(\tau^{\mathrm{real}}, R^{\mathrm{NL}})$.

Translating Natural Language Rules to Code. The refined natural language rules R^{NL} are then translated into executable code using an LLM f_{Code}, yielding the code-based rule set $R^{\mathrm{Code}} = \{r^{\mathrm{Code}}_n\}_{n=1}^{N} = f_{\mathrm{Code}}(R^{\mathrm{NL}})$. See Appendix B.3 for the prompting details. Each rule r^{Code}_n (an example in Step 4 of Figure 3, full examples in Appendix D) returns a success flag (suc^{Code}), feedback, and suggestion. The flag determines transition validity, the feedback and suggestion provide information about the outcome.

Rule Set Pruning via Maximum Coverage. To ensure correctness, each candidate rule is verified against ground-truth transitions from the real trajectory τ^{real}, and any rule that misfires is discarded. For executability, rules that raise errors during any execution are also removed. The rule set is further optimized by selecting those that maximize coverage of incorrectly predicted transitions $\mathcal{D}^{\mathrm{inc}}$, effectively targeting the most critical misalignments. Notably, since the LLM already captures most environment dynamics, the learned rules focus solely on correcting residual errors—resulting in a compact and highly targeted rule set.

As aforementioned, our world model first predicts whether a transition will succeed or fail (binary classification). Hence, our learned rules predict only transition success or failure. Specifically, each rule is formulated as: "if [state matches the condition] then [transition success/failure] otherwise [null]." A rule is activated when the state matches the corresponding conditions, resulting in a non-null output. An activated rule r^{Code}_i that corrects an initially mispredicted transition δ^{inc}_j is said to have covered it.

We formulate rule pruning as a maximum coverage problem, seeking the subset $R^* \subseteq R^{\mathrm{Code}}$:

$$R^* = \underset{R \subseteq R^{\mathrm{Code}}}{\arg\max} \Big| \bigcup_{r^{\mathrm{Code}} \in R} \mathcal{D}^r \Big|, \text{ s.t., } |R| \leq k. \tag{3}$$

where $\mathcal{D}^r$ is the subset of transitions covered by rules, i.e., $\mathcal{D}^r \triangleq \{\delta^{\mathrm{inc}} \in D^{\mathrm{inc}} : \delta^{\mathrm{inc}} \text{ covered by } r^{\mathrm{Code}} \in R^{\mathrm{Code}}\}$. The parameter $k > 0$ is the limit of selected rules, and we find that a large k leads to better performance. We solve this problem using a greedy algorithm (see Appendix 3). This procedure discards rules covering only correct transitions and redundant rules subsumed by broader rules (Step 5 in Figure 3). After pruning, we overwrite the rule set with the selected subset, i.e., we set $R^{\mathrm{Code}} \leftarrow R^*$ and (for simplicity) continue to denote the pruned rule set as R^{Code} in the remainder of the paper. The complete rule learning pipeline described in Section 3.3 is denoted as:

$$R^{\mathrm{Code}} \leftarrow \textsc{RuleLearning}(\tau^{\mathrm{pred}}, \tau^{\mathrm{real}}). \tag{4}$$

3.4 Inference on LLM Agents with Learned Rules

At inference time, code-based rules serve as a post-prediction check to refine the world model's output. The final next state is computed as:

$$\hat{s}_{h+1} = \textsc{Align}(f_{\text{wm}}, R^{\text{Code}}, s_h, a_h), \tag{5}$$

correcting predictions when active rules (i.e., the rule produces a non-null output) detect contradictions. The alignment function is defined as:

$$\hat{s}_{h+1} = \begin{cases} f_{\text{wm}}(s_h, a_h, suc_h^{\text{Code}}), & \text{if } suc_h^{\text{Code}} \neq \hat{suc}_h, \\ f_{\text{wm}}(s_h, a_h, \hat{suc}_h), & \text{otherwise,} \end{cases} \tag{6}$$

where $\hat{suc}_h$ is the success prediction from the world model (see Equation 2), and suc_h^{Code} is the success flag returned by the rule set R^{Code} (Section 3.3). This mechanism allows the model to revise its predictions when symbolic rules detect inconsistencies, improving transition accuracy.

Importantly, we never wholesale replace $\hat{suc}_h$ with suc_h^{Code}. Since the LLM already encodes the bulk of environment dynamics, our rules only correct residual misalignments that the LLM missed. Consequently, the learned rule set remains very compact. Embedding this alignment step into the MPC planner (Algorithm 1) yields more accurate future-state estimates and thus stronger planning performance. Unlike classical MPC, which relies on random sampling, our method leverages the LLM's prior π_{LLM} to propose high-quality candidate actions, significantly reducing the search space.

WALL-EVE operates in a training-free iterative loop alternating between MPC and Rule Learning. During MPC, the LLM agent proposes actions evaluated by a symbolic world model using current rules. In parallel, new rules are learned from recent interactions, and integrated into the model. This continual refinement ultimately enhances its decision-making capabilities (detailed in Algorithm 2).

4 Experiments

4.1 Experimental Setup

Benchmarks. **ALFWorld** is a virtual environment designed as a visual/text-based simulation where agents perform tasks by interacting with a simulated household environment [40]. Text-based refers to the scenario without visual perception, where ground-truth state information is directly included in the input. Visual-based denotes the scenario with visual perception, where ground-truth state information is excluded and agents must perceive visual information using visual perception tools. This benchmark includes six distinct task types, each requiring the agent to accomplish a high-level objective, such as placing a cooled lettuce on a countertop. **Minecraft** is a popular open-world environment. We employ the standard evaluation pipeline provided by MineDojo's TechTree tasks [10], which can be categorized into six levels of increasing difficulty: *Wood*, *Stone*, *Iron*, *Gold*, *Diamond*, and *Redstone*. See Appendix F for details.

Algorithm 1 Model-Predictive Control (MPC)

Input: s_h, R^{Code}, planning horizon H
Initialize: Best plan $a^*_{h:h+H-1} \leftarrow \emptyset$, best state sequence $\hat{s}^*_{h+1:h+H} \leftarrow \emptyset$, best cumulative reward $rew^* \leftarrow -\infty$
Output: a^*_h, $\hat{s}^*_{h+1}$

for all candidate plans $a_{h:h+H-1} \sim \pi_{\text{LLM}}(s_h, H)$ **do**
  $s_t \leftarrow s_h$, $rew \leftarrow 0$
  for $t = h$ to $h + H - 1$ **do**
    $\hat{s}_{t+1} \leftarrow \textsc{Align}(f_{\text{wm}}, R^{\text{Code}}, s_t, a_t)$ /*eq. 5 */
    $rew \leftarrow rew + \mathcal{F}(\hat{s}_{t+1})$
    $\hat{s}_{h+1:t+1}$.append$(\hat{s}_{t+1})$
    $s_t \leftarrow \hat{s}_{t+1}$
  end for
  if $rew > rew^*$ **then**
    $rew^* \leftarrow rew$
    $a^*_{h:h+H-1} \leftarrow a_{h:h+H-1}$
    $\hat{s}^*_{h+1:h+H} \leftarrow \hat{s}_{h+1:h+H}$
  end if
end for

Algorithm 2 WALL-EVE

/* RuleLearning() in Sec. 3.3; MPC() in Sec. 3.4 */
Initialize: $R^{\text{Code}} \leftarrow \emptyset$, $\tau^{\text{real}}, \tau^{\text{pred}} \leftarrow []$, $h \leftarrow 0$, $s_0 \leftarrow \textsc{env}()$

while $\neg\textsc{taskcomplete}(s_h)$ **and** $\neg\textsc{agentdied}(s_h)$ **do**
  $a_h, \hat{s}_{h+1} \leftarrow \text{MPC}(s_h, R^{\text{Code}})$ /*Alg. 1 */
  $s_{h+1} \leftarrow \textsc{env}(a_h)$
  τ^{real}.$\textsc{append}((s_h, a_h, s_{h+1}))$
  τ^{pred}.$\textsc{append}((s_h, a_h, \hat{s}_{h+1}))$
  $R^{\text{Code}} \leftarrow \textsc{RuleLearning}(\tau^{\text{pred}}, \tau^{\text{real}})$ /* eq. 4 */
  $h \leftarrow h + 1$
end while

Baselines. For ALFWorld, the baselines include SOTA training-free methods such as RAFA [30] and AdaPlanner [41], as well as training-based methods such as EMMA [58] and InstructBLIP [8]. For Minecraft, the baselines adopt different action controllers and were originally evaluated in different benchmarks and environments (see Appendix F.2), making direct comparisons infeasible. For fair comparisons, we re-implemented their planning frameworks (since planning is our main focus) in our setting and kept all the other setups the same, as shown in Table 3.

Implementation Details. We adopt different LLM backbones to match the difficulty of each environment: GPT-4o for Minecraft and GPT-3.5-Instruct for ALFWorld. In each environment, we use the same backbone as the corresponding baselines to ensure a fair comparison. For visual perception, Minecraft leverages GPT-4o, while ALFWorld employs a Mask R-CNN model fine-tuned on the

Table 1: Comparison of WALL-EVE with MiniGPT-4 [70], BLIP-2 [26], InstructBLIP [8], EMMA [58], ReAct [61], DEPS [48], AdaPlanner [41], Reflexion [39], RAFA [30] on ALFWorld tasks. ∗-reported in previous work. 'Trained' methods require task-specific training; 'Train-free' methods do not. 'Text environment' includes ground truth inputs; 'Visual environment' requires perception from raw visuals. **Bold** indicates the best result. WALL-EVE lags EMMA in visual mode since EMMA is trained on these tasks, whereas WALL-EVE is training-free.

	Method	Success rate (%) ↑						
		Avg.	Pick	Clean	Heat	Cool	Look	Pick2
	Visual environment							
Trained	MiniGPT-4*	16	4	0	19	17	67	6
	BLIP-2*	4	0	6	4	11	6	0
	InstructBLIP*	22	50	26	23	6	17	0
	EMMA*	**82**	**71**	**94**	**85**	**83**	**88**	**67**
Train-free	ReAct	50	66	31	61	70	32	39
	AdaPlanner	62	83	75	54	63	46	35
	Reflexion	68	79	71	51	**72**	**61**	74
	RAFA	72	86	72	**65**	71	58	77
	WALL-EVE	**75**	**88**	**77**	61	68	**61**	**94**
	Text environment							
Train-free	ReAct*	54	71	65	62	44	28	35
	DEPS*	76	93	50	80	**100**	**100**	0
	AdaPlanner	91	**100**	**100**	89	**100**	97	47
	Reflexion	86	92	94	70	81	90	88
	RAFA	**95**	**100**	97	91	95	**100**	82
	WALL-EVE	**95**	**100**	97	**100**	86	85	**100**

ALFRED dataset [40]. The LLM agent's temperature is set to 0.7, whereas the LLM world-model is run with a temperature of 0.0. Details about the computing resources used for all experiments are provided in Appendix F.

Metrics. (1) **Success rate** (higher is better): the percentage of tasks the agent completes successfully. (2) **Replanning rounds** (lower is better): the number of times the agent revisits the same task to revise its plan for recovering from the failed task planning. (3) **Token cost** (lower is better): the number of tokens consumed by LLM agent/world models during task completion. See Appendix F.1 for details.

4.2 Main Results

WALL-EVE demonstrates superior planning abilities and sample efficiency. In ALFWorld, WALL-EVE achieves the highest success rate while requiring only 17% of the token cost compared to the SOTA method RAFA (Tables 1 and 2).

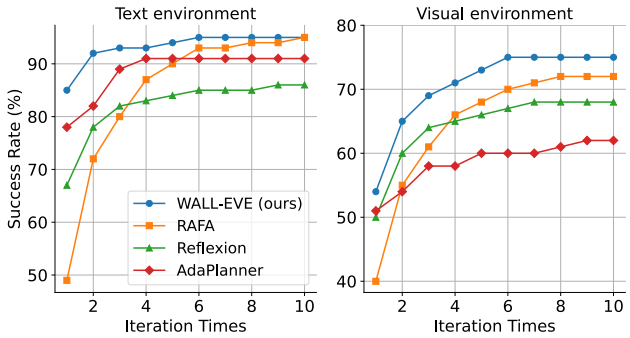

Fig. 4: Comparison of WALL-EVE and baselines on 134 testing tasks from the ALFWorld benchmark in text and visual environments.

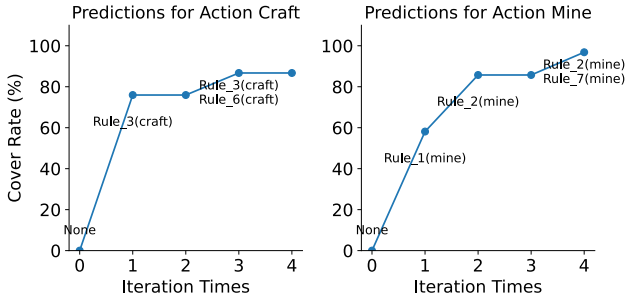

Fig. 5: Cover rate of LLM failed predictions across different actions over iteration times during training. It reflects the proportion of errors corrected by learned rules. Rules added at each iteration are shown under each node.

WALL-EVE also reaches peak performance with fewer iterations (Figure 4). Notably, our method's lower success rate compared to EMMA is expected, as EMMA is trained on test tasks, while ours is training-free. In Minecraft, WALL-EVE significantly outperforms baseline methods by 8–30% in success rate, while using only 60–85% of baseline token usage and requiring 8–20 fewer replanning rounds than other approaches (Tables 3 and 4). Gains are most notable on harder tasks like *Golden*, *Diamond*, and *Redstone*, while model-free methods excel mainly on easier tasks like *Wood* and *Stone*. See Appendix F for additional experimental results, including standard deviations and tests for statistical significance.

Multimodal information processing enhances performance. In Minecraft, as shown in Table 3, the absence of visual modality information leads to a significant performance decline: success rates drop by approximately 10%, and about 5 additional replanning rounds are required. This underscores the necessity of multimodal information processing and highlights the limitations of relying on single-modality data in complex environments.

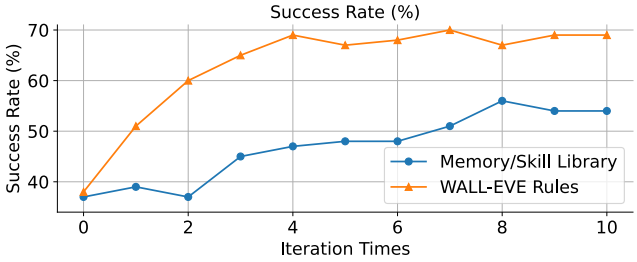

Fig. 6: Learning curve comparison between rule learning (e.g., WALL-EVE) and memory/skill library (e.g., Jarvis-1, Voyager, GITM) over 10 iterations on Minecraft tasks during training.

Table 2: Token-cost comparison on ALFWorld across all baselines. Token cost reports the average number of LLM tokens consumed per task, and Cost reports the corresponding API cost in USD. Lower is better for both metrics, and the lowest cost is highlighted in **bold**, and the second-lowest cost is underlined. WALL-EVE achieves the highest success rate while maintaining the second-lowest token and API cost among all baselines; ReAct is cheaper but attains substantially lower success, whereas RAFA is far more expensive.

Metric	ReAct	AdaPlanner	Reflexion	RAFA	WALL-EVE
Token cost ↓	**53210.94**	74560.55	68727.18	392268.28	68288.96
Cost (USD) ↓	**0.37**	0.52	0.48	2.73	0.47

Visual modality information perception is challenging. Table 1 compares performance across visual and text environments in ALFWorld, with conditions where ground truth information is either excluded or included. The results indicate that all methods experience reduced performance in the visual environment, underscoring the challenges of processing visual information. Contributing factors to this performance reduction include limitations of object detectors, occlusions between objects, and objects falling out of sight. Despite these perceptual challenges, our method, WALL-EVE, consistently exhibits competitive decision-making abilities, highlighting its resilience in visually complex tasks.

4.3 Ablation Study

Rule learning achieves efficient "world alignment". We benchmark our rule-learning approach against memory/skill library methods (e.g., Jarvis-1, Voyager, GITM). To quantify "world alignment," we collect transitions that the LLM world model initially mispredicts and measure the *cover rate*: the fraction of these errors corrected by our learned rules. As shown in Figure 5, both the agent's success rate (Figure 6) and the rule set converge after four iterations, yielding a compact, stable rule library. By iteration four, cover rates exceed

Table 3: Comparison of WALL-EVE with DEPS [48], Plan4MC [2], Jarvis-1 [49], Voyager [45], GITM [71] and Optimus-1 [28] on Minecraft tasks by frameworks, success rate and replanning rounds. Jarvis-1/Voyager/GITM share the same planning framework. Framework components: Reasoning (Rea), Reflector (Ref), Memory (Me), Skill Library (SkLi), Knowledge Graph (KnGr), Rules (Ru). 'w/o visual' removes visual input; 'w/o WM' excludes the world model. Best scores per task are in **bold**.

Method	Planning Framework					Success rate (%) ↑ (Replanning rounds ↓)						
	Rea	Ref	Me/SkLi	KnGr	Ru	Avg.	Wooden	Stone	Iron	Golden	Diamond	Redstone
GPT-4o						9(46.70)	36(35.50)	19(44.72)	0(∞)	0(∞)	0(∞)	0(∞)
DEPS	✓	✓				37(35.36)	83(10.67)	41(33.26)	33(35.27)	22(45.29)	24(42.46)	17(45.22)
Plan4MC	✓		✓			48(28.71)	91(5.77)	75(13.01)	43(34.03)	32(36.59)	26(39.38)	21(43.47)
Jarvis-1/ Voyager/ GITM	✓	✓	✓			54(25.49)	96(3.42)	92(6.01)	57(23.93)	29(37.17)	30(39.80)	22(42.63)
Optimus-1	✓	✓	✓	✓		61(22.45)	95(3.29)	**94**(4.68)	64(21.69)	41(30.49)	31(39.88)	38(34.67)
WALL-EVE w/o visual	✓	✓			✓	59(20.47)	95(2.88)	90(5.00)	62(19.78)	42(29.29)	35(32.50)	33(33.39)
WALL-EVE w/o WM	✓	✓			✓	61(23.13)	94(5.04)	89(9.58)	**67**(**18.56**)	33(39.67)	41(32.73)	43(33.21)
WALL-EVE (ours)	✓	✓			✓	**69**(**15.77**)	**98**(**1.64**)	91(**4.58**)	63(19.38)	**69**(**15.61**)	**46**(**27.08**)	**48**(**26.33**)

Table 4: Comparison of WALL-EVE and baselines on Minecraft tasks for average token cost and API costs (in USD). 'w/o visual' omits visual input, 'w/o WM' excludes the world model. Lowest cost is in **bold**.

Method	Token cost ↓	Cost in USD ↓
DEPS	93560.95	0.65
Jarvis-1/Voyager/GITM	74638.54	0.51
Optimus-1	70566.67	0.48
WALL-EVE w/o visual	78334.69	0.53
WALL-EVE w/o WM	72390.16	0.52
WALL-EVE (ours)	**60348.71**	**0.41**

Table 5: Ablation of WALL-EVE with different configurations on Minecraft tasks. The row in **bold** shows the configuration and performance of WALL-EVE.

Agent	World Model	Success rate (%) ↑	Replanning rounds ↓
LLM	-	37	35.36
LLM	LLM	38	33.53
LLM+rules	-	61	23.13
LLM	**LLM+rules**	**69**	**15.77**
LLM+rules	LLM+rules	67	16.59

87% for `craft` actions and 96% for `mine` actions on the Minedojo benchmark, confirming that our rules efficiently correct residual misalignments.

Rule-aligned world model is critical. Results in Table 5 reveal: (1) Regardless of whether the learned rules are applied within the agent or the world model, adding them significantly enhances the total performance. The success rate increases by 20% to 30% approximately. This observation underscores the crucial role that rules play in improving the effectiveness of WALL-EVE. (2) When

Table 6: Ablation of multimodal perception noise in ALFWorld, reporting success rates (%). Δ reports the performance change when replacing fine-tuned Mask R-CNN with GPT-4o (vision). Best scores are in **bold**.

Perception Model	**Method**			
	AdaPlanner	Reflexion	RAFA	**WALL-EVE**
Fine-tuned Mask R-CNN	62	68	72	**75**
GPT-4o (vision)	48	50	56	**64**
Δ	-14	-18	-16	**-11**

Table 7: Ablation of rule set pruning in Minecraft. **Bold** column indicates the configuration used by WALL-EVE.

Metrics	**w/o Prune**	**w/ Prune - Rule Set Limit**		
		2	4	**no limit (6)**
Success rate (%) ↑	12	52	64	**69**
Replanning rounds ↓	46.79	24.11	18.71	**15.77**

the learned rules are utilized within the world model, they contribute to nearly a 30% improvement in success rate, whereas using rules within the agent result in about a 20% improvement. This disparity may be primarily due to the fact that the learned rules are highly related to the state information (See Appendix D for more details). (3) MPC using a world model without rules offers limited improvement, confirming that the alignment between the world model and the environment dynamics by rule learning is essential for WALL-EVE 's strong performance. (4) "agent+rules, world model+rules" is worse than "agent, world model+rules". Rules are learnt to improve the world model's prediction of environment dynamics. But adding them to agents restricts exploration, leading to entropy collapse, suboptimal MPC sampling, and slightly poorer overall performance. When only the world model uses rules, MPC benefits from rules validation while preserving agent entropy.

Robustness to multimodal perception noise. Table 6 shows that WALL-EVE is more robust to perception noise than strong baselines. In ALFWorld, generic VLMs (e.g., GPT-4o) can misinterpret its non-photorealistic visuals, making an ALFWorld-finetuned Mask R-CNN a stronger perception backbone. When switching from fine-tuned Mask R-CNN to GPT-4o, baseline success drops by $\sim$16% on average, while WALL-EVE decreases by only $\sim$11%. This robustness comes from *logic-level denoising*: rules are induced from aggregated trajectories rather than single frames, so sporadic perception errors rarely form stable, verifiable predicates and are filtered out during rule validation; remaining failures are mainly caused by systematic noise (e.g., persistent mis-detections).

Table 8: World-model prediction fidelity on 500 held-out transitions, comparing the LLM alone with the rule-aligned LLM (LLM+Rules). Success/Failure Acc. evaluates binary transition-outcome prediction; Next-state Attr. Acc. evaluates task-critical attributes in the predicted next state $\hat{s}_{h+1}$; and 5-step Rollout Attr. Acc. measures attribute accuracy after recursively rolling out the world model for five steps. Higher is better for all metrics, and best results are in **bold**.

Metric	Env.	LLM	LLM+Rules
Success/Failure Acc. (%) ↑	ALFWorld	73	**94**
	Minecraft	65	**85**
Next-state Attr. Acc. (%) ↑	ALFWorld	70	**93**
	Minecraft	59	**77**
5-step Rollout Attr. Acc. (%) ↑	Minecraft	31	**68**

Effect of rule pruning. Table 7 shows pruning is essential: without it, noisy/-conflicting rules reduce success and increase replanning. Allowing more rules improves performance, and removing the cap performs best, highlighting pruning's role in keeping a compact, high-impact rule set that aligns the LLM world model with environment dynamics.

World model prediction fidelity beyond binary outcomes. Our world model predicts both whether a transition succeeds or fails and the full next state $\hat{s}_{h+1}$ (Equation 2); the success/failure flag is an intermediate chain-of-thought signal that decomposes the harder direct s_{h+1} prediction. To quantify fidelity beyond binary outcomes, we evaluate 500 held-out transitions (Table 8), where next-state attribute accuracy focuses on task-critical attributes, e.g., inventory in Minecraft and object states/switches in ALFWorld. Rule learning improves next-state attribute accuracy by ∼20% and 5-step rollout accuracy by ∼37%, supporting that our aligned LLM functions as a genuine world model rather than merely a transition verifier.

5 Conclusion

We show that LLMs can act as world models in complex visual environments using our neurosymbolic alignment approach. This method involves two key steps: transforming multimodal inputs into textual representations through Multimodal Parser, and aligning the LLM world model with the environment via rule learning using this textual intermediary. Without needing gradient updates, we bridge the gap between the LLMs' prior knowledge and specific environment dynamics. By integrating this rule-enhanced LLM world model with MPC, our method achieves superior planning abilities and sample efficiency. These findings demonstrate that minimal additional rules can align LLM predictions with environment dynamics, significantly enhancing agent performance in complex visual settings.

References

1. Ali, A., Bai, J., Bala, M., Balaji, Y., Blakeman, A., Cai, T., Cao, J., Cao, T., Cha, E., Chao, Y.W., et al.: World simulation with video foundation models for physical ai. arXiv preprint arXiv:2511.00062 (2025)
2. BAAI, P.: Plan4mc: Skill reinforcement learning and planning for open-world minecraft tasks. arXiv preprint arXiv:2303.16563 (2023)
3. Bain, M., Nagrani, A., Varol, G., Zisserman, A.: Frozen in time: A joint video and image encoder for end-to-end retrieval. In: ICCV (2021)
4. Bruce, J., Dennis, M.D., Edwards, A., Parker-Holder, J., Shi, Y., Hughes, E., Lai, M., Mavalankar, A., Steigerwald, R., Apps, C., et al.: Genie: Generative interactive environments. In: ICML (2024)
5. Carbonneaux, Q., Cohen, G., Gehring, J., Kahn, J., Kossen, J., Kreuk, F., McMilin, E., Meyer, M., Wei, Y., Zhang, D., et al.: Cwm: An open-weights llm for research on code generation with world models. arXiv preprint arXiv:2510.02387 (2025)
6. Chae, H., Kim, N., Ong, K.T.i., Gwak, M., Song, G., Kim, J., Kim, S., Lee, D., Yeo, J.: Web agents with world models: Learning and leveraging environment dynamics in web navigation. arXiv preprint arXiv:2410.13232 (2024)
7. Chi, X., Jia, P., Fan, C.K., Ju, X., Mi, W., Zhang, K., Qin, Z., Tian, W., Ge, K., Li, H., et al.: Wow: Towards a world omniscient world model through embodied interaction. arXiv preprint arXiv:2509.22642 (2025)
8. Dai, W., Li, J., Li, D., Tiong, A., Zhao, J., Wang, W., Li, B., Fung, P., Hoi, S.: InstructBLIP: Towards general-purpose vision-language models with instruction tuning. In: NeurIPS (2023)
9. Du, Y., Yang, S., Dai, B., Dai, H., Nachum, O., Tenenbaum, J., Schuurmans, D., Abbeel, P.: Learning universal policies via text-guided video generation. In: NeurIPS (2024)
10. Fan, L., Wang, G., Jiang, Y., Mandlekar, A., Yang, Y., Zhu, H., Tang, A., Huang, D.A., Zhu, Y., Anandkumar, A.: Minedojo: Building open-ended embodied agents with internet-scale knowledge. In: NeurIPS (2022)
11. Fang, T., Zhang, H., Zhang, Z., Ma, K., Yu, W., Mi, H., Yu, D.: Webevolver: Enhancing web agent self-improvement with coevolving world model. arXiv preprint arXiv:2504.21024 (2025)
12. Gu, Y., Zhang, K., Ning, Y., Zheng, B., Gou, B., Xue, T., Chang, C., Srivastava, S., Xie, Y., Qi, P., et al.: Is your llm secretly a world model of the internet? model-based planning for web agents. arXiv preprint arXiv:2411.06559 (2024)
13. Guan, L., Valmeekam, K., Sreedharan, S., Kambhampati, S.: Leveraging pretrained large language models to construct and utilize world models for model-based task planning. In: NeurIPS (2023)
14. Guo, S., Domingues, O.D., Avalos, R., Courville, A., Strub, F.: World modelling improves language model agents. arXiv preprint arXiv:2506.02918 (2025)
15. Ha, D., Schmidhuber, J.: Recurrent world models facilitate policy evolution. In: NeurIPS (2018)
16. Ha, D., Schmidhuber, J.: World models. arXiv preprint arXiv:1803.10122 (2018)
17. Hafner, D., Lillicrap, T., Ba, J., Norouzi, M.: Dream to control: Learning behaviors by latent imagination. arXiv preprint arXiv:1912.01603 (2019)
18. Hafner, D., Lillicrap, T., Norouzi, M., Ba, J.: Mastering atari with discrete world models. arXiv preprint arXiv:2010.02193 (2020)
19. Hafner, D., Pasukonis, J., Ba, J., Lillicrap, T.: Mastering diverse domains through world models. arXiv preprint arXiv:2301.04104 (2023)

20. Hafner, D., Yan, W., Lillicrap, T.: Training agents inside of scalable world models. arXiv preprint arXiv:2509.24527 (2025)
21. Hao, S., Gu, Y., Ma, H., Hong, J.J., Wang, Z., Wang, D.Z., Hu, Z.: Reasoning with language model is planning with world model. arXiv preprint arXiv:2305.14992 (2023)
22. Hu, A., Russell, L., Yeo, H., Murez, Z., Fedoseev, G., Kendall, A., Shotton, J., Corrado, G.: Gaia-1: A generative world model for autonomous driving. arXiv preprint arXiv:2309.17080 (2023)
23. Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R.H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., et al.: Model-based reinforcement learning for atari. arXiv preprint arXiv:1903.00374 (2019)
24. Kang, B., Yue, Y., Lu, R., Lin, Z., Zhao, Y., Wang, K., Huang, G., Feng, J.: How far is video generation from world model: A physical law perspective. arXiv preprint arXiv:2411.02385 (2024)
25. Kim, S.W., Zhou, Y., Philion, J., Torralba, A., Fidler, S.: Learning to simulate dynamic environments with gamegan. In: CVPR (2020)
26. Li, J., Li, D., Savarese, S., Hoi, S.: Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In: ICML (2023)
27. Li, X., Zhang, Y., Ye, X.: Drivingdiffusion: Layout-guided multi-view driving scene video generation with latent diffusion model. arXiv preprint arXiv:2310.07771 (2023)
28. Li, Z., Xie, Y., Shao, R., Chen, G., Jiang, D., Nie, L.: Optimus-1: Hybrid multimodal memory empowered agents excel in long-horizon tasks. arXiv preprint arXiv:2408.03615 (2024)
29. Lin, Z., Liu, F., Yang, Y., Lyu, J., Gao, Y., Liu, Y., Lu, Z., Yu, Y., Yang, M., Li, J., et al.: Ui-voyager: A self-evolving gui agent learning via failed experience. arXiv preprint arXiv:2603.24533 (2026)
30. Liu, Z., Hu, H., Zhang, S., Guo, H., Ke, S., Liu, B., Wang, Z.: Reason for future, act for now: A principled framework for autonomous llm agents with provable sample efficiency. arXiv preprint arXiv:2309.17382 (2023)
31. Luo, L., Ju, J., Xiong, B., Li, Y.F., Haffari, G., Pan, S.: Chatrule: Mining logical rules with large language models for knowledge graph reasoning. arXiv preprint arXiv:2309.01538 (2023)
32. Mao, X., Lin, S., Li, Z., Li, C., Peng, W., He, T., Pang, J., Chi, M., Qiao, Y., Zhang, K.: Yume: An interactive world generation model. arXiv preprint arXiv:2507.17744 (2025)
33. Matsuo, Y., LeCun, Y., Sahani, M., Precup, D., Silver, D., Sugiyama, M., Uchibe, E., Morimoto, J.: Deep learning, reinforcement learning, and world models. Neural Networks **152**, 267–275 (2022)
34. Micheli, V., Alonso, E., Fleuret, F.: Transformers are sample-efficient world models. arXiv preprint arXiv:2209.00588 (2022)
35. Mu, N., Chen, S., Wang, Z., Chen, S., Karamardian, D., Aljeraisy, L., Hendrycks, D., Wagner, D.: Can llms follow simple rules? arXiv preprint arXiv:2311.04235 (2023)
36. OpenAI: Video generation models as world simulators. Tech. rep., OpenAI (February 2024)
37. Qin, S.J., Badgwell, T.A.: A survey of industrial model predictive control technology. Control engineering practice **11**(7), 733–764 (2003)
38. Robine, J., Höftmann, M., Uelwer, T., Harmeling, S.: Transformer-based world models are happy with 100k interactions. arXiv preprint arXiv:2303.07109 (2023)

39. Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., Yao, S.: Reflexion: Language agents with verbal reinforcement learning. In: NeurIPS (2024)
40. Shridhar, M., Yuan, X., Côté, M.A., Bisk, Y., Trischler, A., Hausknecht, M.: Alfworld: Aligning text and embodied environments for interactive learning. arXiv preprint arXiv:2010.03768 (2020)
41. Sun, H., Zhuang, Y., Kong, L., Dai, B., Zhang, C.: Adaplanner: Adaptive planning from feedback with language models. In: NeurIPS (2024)
42. Tang, H., Key, D., Ellis, K.: Worldcoder, a model-based llm agent: Building world models by writing code and interacting with the environment. arXiv preprint arXiv:2402.12275 (2024)
43. Team, H., Wang, Z., Liu, Y., Wu, J., Gu, Z., Wang, H., Zuo, X., Huang, T., Li, W., Zhang, S., et al.: Hunyuanworld 1.0: Generating immersive, explorable, and interactive 3d worlds from words or pixels. arXiv preprint arXiv:2507.21809 (2025)
44. Tolman, E.C.: Cognitive maps in rats and men. Psychological review **55**(4), 189 (1948)
45. Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., Anandkumar, A.: Voyager: An open-ended embodied agent with large language models. arXiv preprint arXiv:2305.16291 (2023)
46. Wang, X., Zhu, Z., Huang, G., Wang, B., Chen, X., Lu, J.: Worlddreamer: Towards general world models for video generation via predicting masked tokens. arXiv preprint arXiv:2401.09985 (2024)
47. Wang, Y., He, J., Fan, L., Li, H., Chen, Y., Zhang, Z.: Driving into the future: Multiview visual forecasting and planning with world model for autonomous driving. In: CVPR (2024)
48. Wang, Z., Cai, S., Chen, G., Liu, A., Ma, X., Liang, Y., CraftJarvis, T.: Describe, explain, plan and select: interactive planning with large language models enables open-world multi-task agents. In: NeurIPS (2023)
49. Wang, Z., Cai, S., Liu, A., Jin, Y., Hou, J., Zhang, B., Lin, H., He, Z., Zheng, Z., Yang, Y., et al.: Jarvis-1: Open-world multi-task agents with memory-augmented multimodal language models. arXiv preprint arXiv:2311.05997 (2023)
50. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. In: NeurIPS (2022)
51. Wong, L., Mao, J., Sharma, P., Siegel, Z.S., Feng, J., Korneev, N., Tenenbaum, J.B., Andreas, J.: Learning adaptive planning representations with natural language guidance. arXiv preprint arXiv:2312.08566 (2023)
52. Xiang, J., Liu, G., Gu, Y., Gao, Q., Ning, Y., Zha, Y., Feng, Z., Tao, T., Hao, S., Shi, Y., et al.: Pandora: Towards general world model with natural language actions and video states. arXiv preprint arXiv:2406.09455 (2024)
53. Xiang, J., Tao, T., Gu, Y., Shu, T., Wang, Z., Yang, Z., Hu, Z.: Language models meet world models: Embodied experiences enhance language models. In: NeurIPS (2024)
54. Xie, K., Yang, I., Gunerli, J., Riedl, M.: Making large language models into world models with precondition and effect knowledge. arXiv preprint arXiv:2409.12278 (2024)
55. Yang, W., Lin, Y., Zhou, J., Wen, J.R.: Distilling rule-based knowledge into large language models. In: Proceedings of the 31st International Conference on Computational Linguistics (2025)
56. Yang, W., Lin, Y., Zhou, J., Wen, J.: Enabling large language models to learn from rules. arXiv preprint arXiv:2311.08883 (2023)

57. Yang, Y., XU, H., Hu, Z., Yue, Y.: Rlie: Rule generation with logistic regression, iterative refinement, and evaluation for large language models. arXiv preprint arXiv:2510.19698 (2025)
58. Yang, Y., Zhou, T., Li, K., Tao, D., Li, L., Shen, L., He, X., Jiang, J., Shi, Y.: Embodied multi-modal agent trained by an llm from a parallel textworld. In: CVPR (2024)
59. Yang, Z., Li, P., Liu, Y.: Failures pave the way: Enhancing large language models through tuning-free rule accumulation. arXiv preprint arXiv:2310.15746 (2023)
60. Yang, Z., Dong, L., Du, X., Cheng, H., Cambria, E., Liu, X., Gao, J., Wei, F.: Language models as inductive reasoners. In: Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (2024)
61. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., Cao, Y.: React: Synergizing reasoning and acting in language models. In: ICLR (2023)
62. Yu, Y., Yang, M., Li, J., Gao, Y., Liu, F., Yang, Y., Lin, Z., Lyu, J., Liu, Y., Lu, Z., et al.: Proact: Agentic lookahead in interactive environments. arXiv preprint arXiv:2602.05327 (2026)
63. Zhang, Y., Wei, Y., Jiang, D., Zhang, X., Zuo, W., Tian, Q.: Controlvideo: Training-free controllable text-to-video generation. arXiv preprint arXiv:2305.13077 (2023)
64. Zhang, Y., Peng, C., Wang, B., Wang, P., Zhu, Q., Kang, F., Jiang, B., Gao, Z., Li, E., Liu, Y., et al.: Matrix-game: Interactive world foundation model. arXiv preprint arXiv:2506.18701 (2025)
65. Zhao, G., Wang, X., Zhu, Z., Chen, X., Huang, G., Bao, X., Wang, X.: Drivedreamer-2: Llm-enhanced world models for diverse driving video generation. arXiv preprint arXiv:2403.06845 (2024)
66. Zhao, Z., Lee, W.S., Hsu, D.: Large language models as commonsense knowledge for large-scale task planning. In: NeurIPS (2024)
67. Zhen, H., Qiu, X., Chen, P., Yang, J., Yan, X., Du, Y., Hong, Y., Gan, C.: 3d-vla: A 3d vision-language-action generative world model. arXiv preprint arXiv:2403.09631 (2024)
68. Zheng, W., Chen, W., Huang, Y., Zhang, B., Duan, Y., Lu, J.: Occworld: Learning a 3d occupancy world model for autonomous driving. In: ECCV (2025)
69. Zhou, S., Du, Y., Chen, J., Li, Y., Yeung, D.Y., Gan, C.: Robodreamer: Learning compositional world models for robot imagination. arXiv preprint arXiv:2404.12377 (2024)
70. Zhu, D., Chen, J., Shen, X., Li, X., Elhoseiny, M.: Minigpt-4: Enhancing vision-language understanding with advanced large language models. arXiv preprint arXiv:2304.10592 (2023)
71. Zhu, X., Chen, Y., Tian, H., Tao, C., Su, W., Yang, C., Huang, G., Li, B., Lu, L., Wang, X., et al.: Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory. arXiv preprint arXiv:2305.17144 (2023)
72. Zhu, Z., Xue, Y., Chen, X., Zhou, D., Tang, J., Schuurmans, D., Dai, H.: Large language models can learn rules. arXiv preprint arXiv:2310.07064 (2023)