

EVOAGENT: SELF-EVOLVING AGENT WITH CONTINUAL WORLD MODEL FOR LONG-HORIZON TASKS

Tongtong Feng¹, Xin Wang^{1,*}, Zekai Zhou², Ren Wang¹, Yuwei Zhan¹, Guangyao Li¹, Qing Li³, Wenwu Zhu^{1,*}

¹Department of Computer Science and Technology, BNRist, Tsinghua University, China

²Department of Computer Science, University of Sydney, Sydney, Australia

³Department of Electronic Engineering, Tsinghua University, Beijing, China

ABSTRACT

Completing Long-Horizon (LH) tasks in open-ended worlds is an important yet difficult problem for embodied agents. Existing approaches suffer from two key challenges: (1) they heavily rely on experiences obtained from human-created data or curricula, failing to autonomously update and select multimodal experiences, and (2) they may encounter catastrophic forgetting issues when faced with new tasks, failing to autonomously update world knowledge. To solve these challenges, this paper presents *EvoAgent*, a self-evolving agent with a continual World Model (WM), which can autonomously complete various LH tasks across environments through self-planning, self-control, and self-reflection, without human intervention. Our proposed *EvoAgent* contains three modules, i.e., i) the memory-driven planner which uses an LLM along with the WM and interaction memory, to convert LH tasks into executable sub-tasks; ii) the WM-guided action controller which leverages WM to generate low-level actions and incorporates a self-verification mechanism to update multimodal experiences; iii) the experience-inspired reflector which implements a two-stage curriculum learning algorithm to select experiences for task-adaptive WM updates. Moreover, we develop a continual World Model for *EvoAgent*, which can autonomously update the multimodal experience pool and world knowledge through closed-loop dynamics. We conducted extensive experiments on Minecraft and Atair, compared with existing methods, *EvoAgent* can achieve an average success rate improvement of 105% and reduce ineffective actions by more than 6x.

1 INTRODUCTION

Long-horizon (LH) tasks Shen et al. (2025); Guo et al. (2024) are complex, multi-step tasks that require sustained planning, sequential decision-making, and extended execution over a prolonged period to achieve a final goal. These tasks are challenging, often exhibiting reward sparsity Hafner et al. (2025) and procedural diversity Yang et al. (2024). Completing LH tasks in open-ended worlds is an important yet difficult problem for embodied agents, such as logistics robots Luo et al. (2025), surgical robots Marcus et al. (2024), and rescue robots Jadeja et al. (2024).

On the one hand, existing agents have made remarkable progress by utilizing expert data and domain-specific curricula created by humans, developing policies through Reinforcement Learning (RL) Ren et al. (2025); Mazzaglia et al. (2024b), Imitation Learning (IL) Liu et al. (2024), and Large Language Models (LLMs) Li et al. (2025). On the other hand, recent studies Kwa et al. (2025) demonstrate that humans’ ability to accomplish LH tasks in an open world relies on autonomous experience accumulation and world knowledge updates. In essence, autonomous world knowledge update serves as a meta-cognitive driver that not only guides action selection under partial observability but also enables context-aware adaptation to environmental dynamics, thereby resolving the local optimality issue inherent in LH task completion.

Completing long-horizon tasks in open-ended worlds requires embodied agents to achieve autonomous experience accumulation and world knowledge updates, like a baby thrives.

*Corresponding author: Wenwu Zhu and Xin Wang.

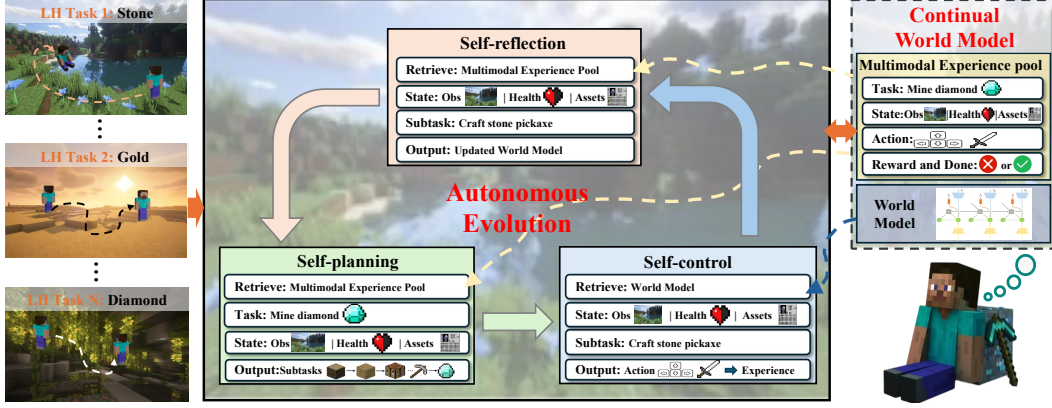


Figure 1: EvoAgent, a self-evolving agent with a continual World Model (WM). Take Minecraft as an example. **Left:** Various Long-Horizon (LH) tasks across environments. **Middle:** EvoAgent can autonomously complete various LH tasks across environments by self-planning, self-control, and self-reflection, without human intervention. **Right:** We build a continual WM for EvoAgent. Through closed-loop dynamics, EvoAgent can autonomously update the multimodal experience pool and world knowledge.

Nevertheless, existing methods are hard to complete various LH tasks across environments from scratch: 1) *Failing to autonomously update and select multimodal experiences.* Most embodied agents assume that all training data are available from the beginning (such as IL-based or LLMs-based agents), which heavily rely on human-created data or curricula Li et al. (2025). However, this assumption is unrealistic, as agents may encounter novel tasks or environments after deployment Zhang et al. (2024). 2) *Failing to autonomously update world knowledge.* Existing methods use LLMs (such as Voyager Wang et al. (2023a), Jarvis-1 Wang et al. (2023c)) to represent world knowledge based on sampling historical experiences or use a graph (such as Optimus-1 Li et al. (2024)) to sparsely represent world knowledge, which requires human intervention and is hard to autonomously update. Existing methods face catastrophic forgetting, where they lose previously obtained knowledge Nayak et al. (2025); Hafner et al. (2025) for learning new tasks, which are hard to autonomously update and transfer world knowledge for LH tasks across environments.

To solve this problem, in this paper, we propose **EvoAgent** (as shown in Figure 1), a self-evolving agent with a continual World Model (WM), which can autonomously complete various LH tasks across environments through self-planning, self-control, and self-reflection, without human intervention. Our proposed EvoAgent contains three modules: i) The experience-driven task planner, which uses an LLM along with interaction experiences, to incorporate self-state into the planning phase and convert LH tasks into executable sub-tasks; ii) The WM-guided action controller, which leverages WM to generate low-level actions and incorporates a self-verification mechanism to update multimodal experiences. iii) The Curriculum Learning (CL)-based reflector, which implements a two-stage CL algorithm to select experiences for task-adaptive WM updates. Moreover, we propose a novel continual WM for EvoAgent as well. By utilizing a model-based online RL setup and closed-loop dynamics, EvoAgent is able to autonomously update the multimodal experience pool and world knowledge, filtering out invalid explorations and mitigating historical forgetting.

We evaluate EvoAgent’s performance in Minecraft Fan et al. (2022), a popular open-world environment. Extensive experiments demonstrate EvoAgent’s superiority: compared with existing methods, EvoAgent can achieve an average success rate improvement of 105% and reduce ineffective actions by more than 6x. Ablation studies confirm that our Continual WM contributes 72% of the performance gain by enabling coherent knowledge integration. We also evaluate the generalization of EvoAgent in the Atari environment Bellemare et al. (2013). The contributions of this paper are summarized as follows:

- We propose EvoAgent, which can autonomously complete various LH tasks across various environments through self-planning, self-control, and self-reflection, without human intervention.

- We build a novel continual WM for EvoAgent, which can autonomously update the multi-modal experience pool and world knowledge through closed-loop dynamics.
- We conduct extensive experiments on Minecraft and Atair to validate the superiority of EvoAgent, where the proposed EvoAgent can achieve an average success rate improvement of 105% and reduce ineffective actions by more than 6x compared with existing methods.

2 RELATED WORKS

Embodied agents solving long-horizon tasks. Long-Horizon (LH) tasks Shen et al. (2025); Guo et al. (2024); Chen et al. (2024) refer to complex, multi-step tasks. Existing work on embodied agents completing LH tasks can be divided into two categories. One is Model-Based Reinforcement Learning (MBRL) Mazzaglia et al. (2024a). Embodied agents leverage MBRL to tackle LH tasks by interacting with environments and learning predictive world dynamics Liu et al. (2024). Such as GenRL Mazzaglia et al. (2024b) proposes a multimodal-foundation model that aligns vision-language representations with generative world dynamics for RL. The other is vision-language model-based (VLM) planning Roger et al. (2025). Embodied agents leverage VLMs to decompose LH tasks into hierarchical sub-goals Liu et al. (2024), dynamically refine plans via memory-augmented reasoning Song et al. (2024), and align semantic intent with executable actions through iterative simulation Yang et al. (2024), such as EmbodiedGPT Mu et al. (2023), which bridges high-level planning with low-level control. However, they assume perfect knowledge of environments, rely on oracle feedback, and assume perfect execution of low-level policies, which makes it hard to adapt various LH tasks across environments in open worlds Zhang et al. (2024).

World Model (WM). WMs are foundational blocks of AI systems to perform planning and reasoning Ha & Schmidhuber (2018). They serve as simulators of real environments that predict the future outcome of certain actions, and policies can be derived from them. Current research focuses on two paradigms: understanding the world through latent state representations Hansen et al. (2023); Zhou et al. (2024) and predicting future dynamics for planning and control Ma et al. (2024); Wang et al. (2024). Representative example usages of them in MBRL include action searching Nayak et al. (2025); Schrittwieser et al. (2020), policy optimization within such simulators Feinberg et al. (2018); Hafner et al. (2019a), or a combination of both Hafner et al. (2025); Chitnis et al. (2023). However, WMs currently struggle to prevent catastrophic forgetting Mattes et al. (2023) due to their inability to maintain stable representations of previously learned environmental dynamics while adapting to new tasks, often exacerbated by shared parameter updates prior to knowledge Sun et al. (2024).

3 EVOAGENT

Framework. Let \mathcal{E} denote a dynamic open-world environment with partial observability, \mathcal{T} represent the long-horizon tasks, and \mathcal{S} represents the agent’s current state. We aim to design a self-evolving agent *EvoAgent* that can complete various long-horizon tasks across environments, without human intervention. As shown in Figure 2, EvoAgent includes an experience-driven task planner Ψ_{plan} , a WM-guided action controller Π_{act} , a CL-based reflector Φ_{reflect} ; The continual world model includes a Multimodal Experience Pool (MEP) \mathcal{D}_{MEP} , and a world model \mathcal{M}_w . EvoAgent can be instantiated as:

$$\text{EvoAgent} : \langle \Psi_{\text{plan}}, \Pi_{\text{act}}, \Phi_{\text{reflect}}, \mathcal{D}_{\text{MEP}}, \mathcal{M}_w \rangle \quad (1)$$

Continual world model. As shown in Algorithm 1 at Appendix A, EvoAgent with a continual world model can self-evolve through closed-loop dynamic self-planning, self-control, and self-reflection, autonomously updating the multimodal experience pool and world knowledge, filtering invalid exploration, mitigating historical forgetting, and contributing to better autonomous planning and action control. The sketch of EvoAgent is as follows:

$$\begin{array}{c} \mathcal{E}, \mathcal{T}, \mathcal{S}, \mathcal{D}_{\text{MEP}}, \mathcal{M}_w \\ \text{Planner} \rightarrow \text{Controller} \rightarrow \text{Reflector} \rightarrow \\ \Psi_{\text{plan}} \triangleright \mathcal{D}_{\text{MEP}} \quad \Pi_{\text{act}} \circ \mathcal{M}_w \quad \Phi_{\text{reflect}} \triangleright \mathcal{D}_{\text{MEP}} \\ \downarrow \{g_i\} \quad \downarrow \{a_t\}, \mathcal{D}_{\text{MEP}} \quad \downarrow \theta', \mathcal{M}_w \end{array} \quad (2)$$

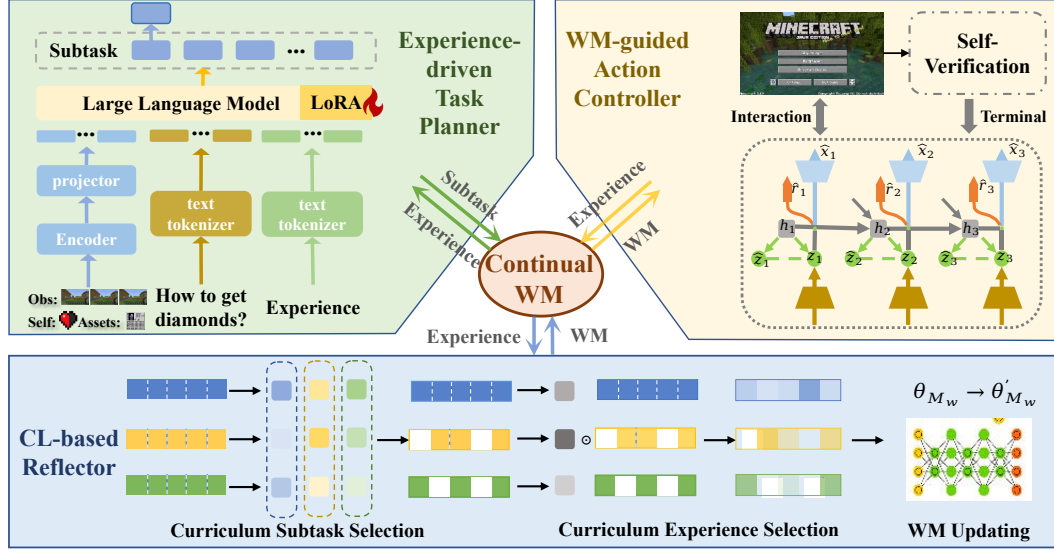


Figure 2: EvoAgent Framework, which includes three modules empowered by a continual WM.

where $\{g_i\}$ are subtasks generated by the planner Ψ_{plan} ; $\{a_t\}$ are actions generated by the controller Π_{act} ; $\theta'_{\mathcal{M}_w}$ is the updated parameter of the world model \mathcal{M}_w .

Evaluation. According to relevant research Hafner et al. (2025); Guo et al. (2024), the agents' performance evaluation includes Success Rate (SR) and Exploration Efficiency (EE).

$$SR = \frac{\text{Episode}_{g_i}^{\text{suc}}}{\text{Episode}^{\text{all}}}, EE = \frac{\mathcal{L}_{g_i}^{\text{suc}}}{\mathcal{L}_{g_i}^{\text{all}}} \quad (3)$$

where $\text{Episode}_{g_i}^{\text{suc}}$ indicates the number of episodes in which the subtask g_i succeeded; $\text{Episode}_{g_i}^{\text{all}}$ indicates the total number of episodes; $\mathcal{L}_{g_i}^{\text{suc}}$ indicates the success step length of subtask g_i , and $\mathcal{L}_{g_i}^{\text{all}}$ indicates the total step length of subtask g_i exploration.

3.1 PRELIMINARIES

World model. Recurrent State-Space Model (RSSM) Hafner et al. (2025; 2019b) is a classic world model structure, which can predict latent states and rewards from high-dimensional observations. RSSM contains 6 modules. 1) Encoder, maps observation o_t to a stochastic latent state $s_t = (h_t, z_t)$, where h_t is a deterministic RNN state and z_t is a stochastic latent variable, $q_\phi(z_t|h_t, o_t) = \mathcal{N}(z_t; \mu_\phi(h_t, o_t), \sigma_\phi(h_t, o_t))$, where μ_ϕ, σ_ϕ are neural networks. 2) Sequence model: predicts the sequence of these representations given past actions a_{t-1} , $h_t = f_\theta(h_{t-1}, z_{t-1}, a_{t-1})$. 3) Dynamics predictor, predicts the prior latent state transition, $p_\theta(\hat{z}_t|h_t) = \mathcal{N}(\hat{z}_t; \mu_\theta(h_t), \sigma_\theta(h_t))$. 4) Decoder: reconstructs observations from latent states, $p_\theta(o_t|h_t, z_t) = \mathcal{N}(o_t; \mu_\theta^{\text{obs}}(h_t, z_t), \sigma_\theta^{\text{obs}})$. 5) Reward predictor, predicts rewards, $\hat{r}_t = r_\theta(h_t, z_t)$. 6) Continual predictor, predicts episode continuation flags, $\hat{c}_t = \text{sigmoid}(c_\theta(h_t, z_t))$. Above all, RSSM can be defined as follows:

$$\text{Encoder:} \quad z_t \sim q_\phi(z_t|h_t, o_t) \quad (4)$$

$$\text{Sequence model:} \quad h_t = f_\theta(h_{t-1}, z_{t-1}, a_{t-1}) \quad (5)$$

$$\text{Dynamics predictor:} \quad \hat{z}_t \sim p_\theta(\hat{z}_t|h_t) \quad (6)$$

$$\text{Decoder:} \quad \hat{o}_t \sim p_\theta(\hat{o}_t|h_t, z_t) \quad (7)$$

$$\text{Reward predictor:} \quad \hat{r}_t \sim r_\theta(\hat{r}_t|h_t, z_t) \quad (8)$$

$$\text{Continual predictor:} \quad \hat{c}_t \sim c_\theta(\hat{c}_t|h_t, z_t) \quad (9)$$

Online model-based reinforcement learning. As shown in Appendix B.

3.2 EXPERIENCE-DRIVEN TASK PLANNER

The experience-driven task planner Ψ_{plan} is formalized as a function that maps the current multimodal state \mathcal{S} , long-horizon task \mathcal{T} , and experience \mathcal{D}_{MEP} to a sequence of subtasks \mathcal{G} .

$$\Psi_{\text{plan}} : \mathcal{S} \times \mathcal{T} \times \mathcal{D}_{\text{MEP}} \rightarrow \mathcal{G} \quad (10)$$

$$\mathcal{S} = \mathcal{O}_{\text{obs}} \times \mathcal{S}_{\text{self}} \times \mathcal{S}_{\text{assets}}, s_t \in \mathcal{S} \quad (11)$$

$$\mathcal{D}_{\text{MEP}} = \{h\}, h = \langle (s_t, a_t, r_t, s_{t+1}), \mathbb{P}_{(g_i)} | g_i \rangle \quad (12)$$

where $\mathcal{G} = \{g_i\}_{i=1}^n$ is the subtask space, each subtask g_i satisfies $\bigcup_{i=1}^n g_i \supseteq \mathcal{T}$; \mathcal{O}_{obs} represents first-person observations, $\mathcal{S}_{\text{self}}$ represents the agent’s internal state, such as health or hunger, and $\mathcal{S}_{\text{assets}}$ represents agent’s asset library, such as tools; s_t represents multimodal state at step t ; h represents the experience; r_t represents the reward obtained by performing action a_t at state s_t ; $\mathbb{P}_{(g_i)}$ indicates the percentage of subtask g_i completion.

As shown in Figure 2, we adopt the image tokenizer f_v to encode the raw images $\mathcal{O}_{\text{obs}}, \mathcal{S}_{\text{self}}, \mathcal{S}_{\text{assets}}$ into token embeddings $\mathcal{V} = \{v_1, v_2, \dots, v_n\} \in \mathbb{R}^{n \times d}$, where n denotes the number of visual tokens and d is the dimensionality of each token. We adopt the textual tokenizer f_t to encode \mathcal{T} and \mathcal{D}_{MEP} into token embeddings. We further utilize a lightweight projection module f_l with a trainable projection matrix W . This module maps the visual tokens to the same space with text embeddings $\hat{\mathcal{V}} = W\mathcal{V}$, yielding $\hat{\mathcal{V}} = \{\hat{v}_1, \hat{v}_2, \dots, \hat{v}_n\} \in \mathbb{R}^{n \times d}$.

The general LLM backbone Θ of our planner is built upon a decoder-only architecture with causal transformers. Our model employs an auto-regressive prediction mechanism, generating responses based on the provided multimodal input tokens. The resulting response is a mixed sequence of visual and textual tokens, represented as $Z = \{z_1, z_2, \dots, z_m\}$. For each embedding z_i , we pass it through a linear layer f_p followed by a softmax operation, mapping it into a probability distribution of the language vocabulary. The final subtask g_i for the i -th token z_i is determined by selecting the token from the existing language codebook with the highest score: $g_i = \text{argmax}(\text{softmax}(f_p(z_i)))$.

When the WM-guided action controller feedback indicates task failure, the LLM-based planner undergoes lightweight fine-tuning using Low-Rank Adaptation (LoRA) Hu et al. (2021) to enhance its adaptability to encountered challenges. The fine-tuning process utilizes all accumulated experiences from the multimodal experience pool \mathcal{D}_{MEP} as training data. Specifically, after a subtask g_i fails, experience trajectories relevant to the subtask g_i are extracted to construct input-output pairs $\{(X_{\text{in}}^{(k)}, X_{\text{out}}^{(k)})\}$, where the input includes all the experience h related the subtask g_i , while the output represents the corresponding subtask sequence. LoRA adapters are injected into the attention modules of the LLM, and only these low-rank parameters are updated during training, keeping the original LLM parameters frozen. The optimization objective is to maximize the cross-entropy loss between the predicted and history subtask sequences. This enables the planner to quickly study from the failure patterns while preserving its general planning capabilities, thereby improving robustness and reducing repeated errors in long-horizon tasks.

3.3 WM-GUIDED ACTION CONTROLLER

The WM-guided action controller Π_{act} is formalized as a function that maps the current multimodal state \mathcal{S} , subtask \mathcal{G} , and the world model \mathcal{M}_w to an action sequence $a_{t:t+H} = \{a_t, a_{t+1}, \dots, a_{t+H}\}$ for horizon H .

$$\Pi_{\text{act}} : \mathcal{S} \times \mathcal{G} \times \mathcal{M}_w \rightarrow \mathcal{A} \quad (13)$$

Action selection. The controller utilizes \mathcal{M}_w to predict future states and optimize actions:

$$a_{t:t+H} = \arg \max_{a_{t:t+H} \in \mathcal{A}^H} \mathbb{E}_{\mathcal{M}_w} \left[\sum_{\tau=t}^{t+H} \gamma^{\tau-t} R(s_\tau, a_\tau, g_i) \right] \quad (14)$$

where $R(s_\tau, a_\tau, g_i)$ is the goal-aligned reward function, and $\gamma \in [0, 1]$ is the discount factor.

Self-verification. After executing a_t , the agent interacts with \mathcal{E} to collect environment feedback. Then it uses a self-verification mechanism to determine whether the subtask g_i can be terminated.

Let $\phi_{\text{verify}} : \mathcal{S} \times \mathcal{G} \times \mathbb{T} \rightarrow \{\text{Terminal}, \text{N-Terminal}\}$ denote the self-verification module, where:

$$\phi_{\text{verify}}(s_t, g_i, t) = \begin{cases} \text{Terminal} & \text{if } \cos(\text{Emb}_{s_t}, \text{Emb}_{g_i}) \geq \sigma \vee t \geq T_{\max} \\ \text{N-terminal} & \text{otherwise} \end{cases} \quad (15)$$

where Emb_{s_t} is the WM-encoded latent representation of the current state, Emb_{g_i} is the task embedding derived from the subtask description. The similarity function $\cos(\cdot)$ represents the cosine similarity, which is used to measure goal attainment with threshold σ set empirically. T_{\max} is the maximum allowed steps of each episode. When a subtask g_i is completed or the subtask g_i completion cycle exceeds the maximum step length T_{\max} , the subtask g_i is terminated and the experience-driven task planner is performed again.

MEP Updating. If the subtask g_i is terminated, whether it is successful or exceeds the step threshold, $\{s_t, a_t, r_t, s_{t+1}, \mathbb{P}_{(g_i)}|g_i\}_{t=0}^T$ is added to the multimodal experience pool \mathcal{D}_{MEP} .

$$\mathcal{D}_{\text{MEP}} \leftarrow \mathcal{D}_{\text{MEP}} \cup \{s_t, a_t, r_t, s_{t+1}, \mathbb{P}_{(g_i)}|g_i\}_{t=0}^T \quad (16)$$

New experiences will be autonomously added to the multimodal experience pool.

3.4 CL-BASED REFLECTOR

The CL-based reflector Φ_{reflect} is formalized as a function that maps the current multimodal state \mathcal{S} , subtask \mathcal{G} , and the multimodal experience \mathcal{D}_{MEP} to update the world model from \mathcal{M}_w to \mathcal{M}'_w .

$$\Phi_{\text{reflect}} : \mathcal{S} \times \mathcal{G} \times \mathcal{D}_{\text{MEP}} \times \mathcal{M}_w \rightarrow \mathcal{M}'_w \quad (17)$$

Φ_{reflect} employs a two-stage CL algorithm to optimize experience selection. Then Φ_{reflect} update the world model to improve exploration efficiency, and autonomously updates world knowledge.

A. TWO-STAGE CL ALGORITHM

Stage 1: curriculum subtask selection. For candidate subtasks $g_i \in \mathcal{G}$, we use four indicators for curriculum subtask selection: (1) the relevance of the subtask g_i to the current target task $\mathcal{T}_{\text{goal}}$; (2) the exploration efficiency of the subtask g_i (ratio of successful step length $L_{g_i}^{\text{suc}}$ to total step length $L_{g_i}^{\text{all}}$); (3) the importance of the subtask g_i (comparing its impact on the current world model $\mathcal{M}_{w,g_i}^{\text{new}}$ and past world model $\mathcal{M}_{w,g_i}^{\text{old}}$); (4) the completion ratio of the subtask $\mathbb{P}_{(g_i)}$.

Therefore, curriculum subtask g_i priority score $\rho(g_i)$ for experience $h = \langle s_t, a_t, r_t, s_{t+1}, \mathbb{P}_{(g_i)}|g_i \rangle$ can be defined as follows:

$$\begin{aligned} \tau(g_i) = & \underbrace{\lambda_1 \cdot \cos(\text{Emb}_{g_i}, \text{Emb}_{\mathcal{T}_{\text{goal}}})}_{\text{Relevance}} + \underbrace{\lambda_2 \cdot \frac{L_{g_i}^{\text{suc}}}{L_{g_i}^{\text{all}}}}_{\text{Efficiency}} \\ & + \underbrace{\lambda_3 \cdot \text{KL}(\mathcal{M}_{w,g_i}^{\text{old}} \parallel \mathcal{M}_{w,g_i}^{\text{new}})}_{\text{Importance}} + \underbrace{\lambda_4 \cdot \mathbb{P}_{(g_i)}}_{\text{Completion ratio}} \end{aligned} \quad (18)$$

where $\cos(\text{Emb}_{g_i}, \text{Emb}_{\mathcal{T}_{\text{goal}}})$ represents the cosine similarity of task embedding. $\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1$ are balancing coefficients. Finally, in round k , $|\mathcal{D}_k^{\text{subtask}}|$ subtasks are selected.

$$\mathcal{D}_k^{\text{subtask}} = \{g_i | \tau(h_i) \geq \rho_k\}, \quad \rho_k = \rho_0 \cdot e^{-c_s k} \quad (19)$$

with c_s controlling curriculum subtask progression rate.

Stage 2: curriculum experience selection. For candidate experience $h \in \mathcal{D}_{\text{MEP}}$ in selected subtasks $\mathcal{D}_k^{\text{subtask}}$, we use three indicators for curriculum experience selection: (1) the Temporal Difference Error (TD-Error) $\delta_{\text{TD}}(h_j)$, prioritizes experience with high TD-Error, indicating prediction mismatch between current and target world models; (2) the Gradient Norm $\|\nabla_{\mathcal{M}_w} \mathcal{L}_{\text{pred}}(h_j)\|$, favors experiences that maximally influence the world model's parameter updates; (3) the Information Gain, measures how much the experience h_j changes the world model's belief distribution, calculated via KL divergence between current $\mathcal{M}_w^{\text{new}}(s_{j+1}|h_j)$ and previous $\mathcal{M}_w^{\text{old}}(s_{j+1}|h_j)$ world model

predictions.

$$\begin{aligned} \epsilon(h_j) = & \underbrace{\eta_1 \cdot |\delta_{\text{TD}}(h_j)|}_{\text{TD-Error}} + \underbrace{\eta_2 \cdot \|\nabla_{\mathcal{M}_w} \mathcal{L}_{\text{pred}}(h_j)\|_2}_{\text{Gradient Norm}} \\ & + \underbrace{\eta_3 \cdot \text{KL}(\mathcal{M}_w^{\text{new}}(s_{j+1}|h_j) \parallel \mathcal{M}_w^{\text{old}}(s_{j+1}|h_j))}_{\text{Information Gain}} \end{aligned} \quad (20)$$

where $\eta_1 + \eta_2 + \eta_3 = 1$ are balancing coefficients. Finally, in round k , $|\mathcal{D}_k^{\text{exp}}|$ experiences are selected.

$$\mathcal{D}_k^{\text{exp}} = \{h_j | \epsilon(h_j) \geq \rho_k\}, \quad \rho_k = \rho_0 \cdot e^{-c_h k} \quad (21)$$

with c_h controlling curriculum experience progression rate.

B. WORLD MODEL UPDATING

Update the world model \mathcal{M}_w using experiences $\mathcal{D}_k^{\text{exp}}$ with importance-aware weight w_j :

$$\theta'_{\mathcal{M}_w} \leftarrow \theta_{\mathcal{M}_w} - \nabla \left[\underbrace{\sum_{h_j} w_j \mathcal{L}_{\text{pred}}(h_j)}_{\text{Curriculum Loss}} + \underbrace{\mu \cdot \Omega(\theta, \theta^{\text{old}})}_{\text{Regularization}} \right] \quad (22)$$

$$w_j = \frac{\epsilon(h_j)}{\max_k \epsilon(h_k)}, \Omega = \sum_i \mathcal{F}_i (\theta_i - \theta_i^{\text{old}})^2 \quad (23)$$

where w_j to emphasize critical experiences, and Ω to penalize shifts in parameters critical for past tasks. \mathcal{F}_i is the Fisher information matrix diagonal.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETTING

Simulators. We use Minecraft Fan et al. (2022) to evaluate EvoAgent. Minecraft features a procedurally generated 3D world of different biomes, which consists of 1-meter-sized blocks that the player can break and place. There are about 30 different creatures that the player can interact with or fight. We employ MineRL 0.4.4 with Minecraft as our simulation environment. The agent operates at a fixed speed of 20 frames per second and only interacts with the environment via low-level control signals. Optimus-1 Li et al. (2024) constructs a benchmark of 67 tasks to evaluate the Agent’s ability for long-horizon tasks. We use the same task group partitioning as the Optimus-1 to evaluate EvoAgent. We also test the cross-environment generalization of our method in the Atari simulator. *Atari* Bellemare et al. (2013) is a cutting-edge, high-fidelity simulation environment for multi-physics analysis and hardware-in-the-loop testing in aerospace and robotics.






Hyperparameters. EvoAgent is designed based on the codebase of dreamerV3 Hafner et al. (2025). The planner of EvoAgent uses the VQ-GAN (Esser et al., 2021) and GPT-4o for task planning. The controller of EvoAgent uses the RSSM-based WM Hafner et al. (2025) for action selection. EvoAgent runs on a single A100 GPU. Taking 10^7 steps as an example, compared to dreamerV3 running for 7 days, EvoAgent only needs to run for 2.7 days. For detailed hyperparameters, please refer to the Appendix C.

Baselines. We compare EvoAgent with existing outperforming agents, including WM-based agents (such as PPO Schulman et al. (2017), dreamerV3 Hafner et al. (2025)) and LLM-based agents (such as GPT-4V, Jarvis-1 Wang et al. (2023d), Optimus-1 Li et al. (2024)) on the challenging long-horizon tasks cross-environments. Note that we initialize all agents with an empty multimodal experience pool, while PPO and Jarvis-1 have tools in their initial state. We do not consider agents that are completely based on human data and curricula support (such as Voyager Wang et al. (2023a), DEPS Wang et al. (2023b), Steve-Eye Zheng et al. (2023), and Plan4MC BAAI (2023)).

4.2 EXPERIMENTAL RESULTS

As shown in Table 1, EvoAgent achieves state-of-the-art success rates (SR) and exploration efficiency (EE) across all resource tiers. Compared with existing methods, EvoAgent can achieve an

Table 1: Main result of EvoAgent. We report the average success rate (SR) and average exploration efficiency (EE) on each task group (as shown in Eq. 3). Upper EE metrics mean that the agent is more efficient at completing the task with fewer invalid exploration steps, while 0.00 indicates that the agent is unable to complete the task. The Overall represents the average result on the three groups of Iron, Gold, and Diamond. The Improving represents the average performance improvement of EvoAgent compared to the algorithms Jarvis-1, dreamerV3, and Optimus-1.

| Group | Metric | PPO | GPT-4V | Jarvis-1 | dreamerV3 | Optimus-1 | EvoAgent | Improving (%) |
|-------------------------------------------------------------------------------------------|--------|-------|--------|----------|-----------|-----------|--------------|---------------|
|  Wood | SR↑ | 28.16 | 35.24 | 89.73 | 91.07 | 96.39 | 97.47 | 5.49 |
| | EE↑ | 53.82 | 69.45 | 87.36 | 93.22 | 97.82 | 98.43 | 6.07 |
|  Stone | SR↑ | 13.42 | 14.39 | 81.91 | 86.82 | 88.79 | 94.53 | 10.12 |
| | EE↑ | 27.56 | 30.64 | 84.72 | 88.39 | 89.25 | 96.48 | 10.32 |
|  Iron | SR↑ | 0.00 | 0.00 | 42.38 | 33.79 | 45.48 | 51.82 | 27.79 |
| | EE↑ | 0.00 | 0.00 | 47.52 | 35.68 | 46.16 | 58.54 | 35.76 |
|  Gold | SR↑ | 0.00 | 0.00 | 8.84 | 6.57 | 10.62 | 21.69 | 149.98 |
| | EE↑ | 0.00 | 0.00 | 9.76 | 8.05 | 8.03 | 30.48 | 253.87 |
|  Diamond | SR↑ | 0.00 | 0.00 | 7.69 | 4.73 | 9.30 | 17.36 | 139.77 |
| | EE↑ | 0.00 | 0.00 | 0.07 | 3.69 | 7.31 | 26.83 | 627.10 |
| Overall | SR↑ | 0.00 | 0.00 | 19.64 | 15.03 | 21.80 | 30.29 | 105.85 |

average success rate improvement of 105% and reduce ineffective actions by more than 6x. For basic tasks (Wood/Stone), EvoAgent marginally outperforms Optimus-1 (97.47% vs. 96.39% SR on Wood) but exhibits significantly greater advantages in advanced tasks like Gold (21.69% vs. 10.62% SR) and Diamond (17.36% vs. 9.30% SR). This hierarchy-aligned improvement suggests EvoAgent’s closed-loop planning-control-reflection mechanism effectively addresses long-horizon dependencies, where traditional model-based methods (DreamerV3) and LLM-driven agents (Jarvis-1) struggle to maintain coherent multi-stage strategies. Notably, the EE metric reveals EvoAgent’s exploration superiority: its 30.48% EE on Gold tasks is 3.8× higher than Optimus-1, indicating drastically reduced invalid actions during deep resource acquisition.

Model-free methods (PPO) and pure vision-language models (GPT-4V) fail completely (0% SR/EE) on tasks requiring tool hierarchies (Iron+), highlighting their inability to model latent state transitions. While Jarvis-1 and DreamerV3 achieve partial success on intermediate tasks (42.38% SR on Iron), their performance collapses on Gold/Diamond tiers due to compounding errors in action sequences. The 26.83% EE for EvoAgent on Diamond tasks, 7.3× higher than Optimus-1, underscores how CL-based experience selection mitigates exploration bottlenecks in sparse-reward scenarios. This aligns with our hypothesis that conventional prioritization heuristics inadequately handle exponentially growing action spaces in long-horizon tasks.

We also test the cross-environment generalization of our method in the Atair simulator. The experimental results are shown in the Appendix D.

4.3 ABLATION STUDY

The ablation study reveals critical insights into the contributions of individual components (Planning without LoRA, Planning with LoRA, Control, Reflection only with stage 1, Reflection only with stage 2, Reflection with both stages) and Continual WM to LH tasks. We selected 10 random seeds for testing. Table 2 shows the mean and variance of the average success rate (SR) for each ablation study. When only PPO is used without any modules (first row), the agent fails to progress beyond basic tasks (28.16% SR for Wood, 0% for Iron+). Introducing the Planning module nearly doubles performance on Wood (45.69%) and marginally improves Stone (18.37%), but still fails to unlock advanced tasks (Iron+ at 0%), suggesting that planning alone cannot resolve the exploration bottleneck in LH tasks. A pivotal leap occurs when Control is added (P+C), with Wood and Stone success rates surging to 92.42% and 85.31%, respectively, and modest progress in Iron (31.59%). This underscores the necessity of structured exploration to navigate intermediate dependencies. However,

Table 2: Ablation study results. We report the average success rate (SR) on each task group. P⁻, P⁺, C⁺, R¹, R², R⁺, and CWM represent Planning without LoRA, Planning with LoRA, Control, Reflection only with stage 1, Reflection only with stage 2, Reflection with both stages, and Continual World Model, respectively. The PPO algorithm is used by default for model decision-making.

| Setting | | | | | | | Tasks | | | | |
|----------------|----------------|----------------|----------------|----------------|----------------|-----|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|
| P ⁻ | P ⁺ | C ⁺ | R ¹ | R ² | R ⁺ | CWM | Wood | Stone | Iron | Gold | Diamond |
| ✓ | | | | | | | 28.16 \pm 6.01 | 13.42 \pm 7.62 | 0.00 \pm 0.00 | 0.00 \pm 0.00 | 0.00 \pm 0.00 |
| | | | | | | | 41.36 \pm 5.20 | 16.27 \pm 8.32 | 0.00 \pm 0.00 | 0.00 \pm 0.00 | 0.00 \pm 0.00 |
| | ✓ | | | | | | 45.69 \pm 5.12 | 18.37 \pm 6.71 | 0.00 \pm 0.00 | 0.00 \pm 0.00 | 0.00 \pm 0.00 |
| | ✓ | ✓ | | | | | 92.42 \pm 3.31 | 85.31 \pm 5.96 | 31.59 \pm 6.72 | 5.47 \pm 2.46 | 3.52 \pm 2.31 |
| | ✓ | ✓ | ✓ | | | | 93.18 \pm 2.86 | 87.72 \pm 5.74 | 34.63 \pm 5.19 | 8.68 \pm 2.72 | 5.14 \pm 2.51 |
| | ✓ | ✓ | | ✓ | | | 95.37 \pm 2.48 | 91.26 \pm 3.86 | 39.58 \pm 5.08 | 14.20 \pm 4.05 | 8.93 \pm 3.73 |
| | | | | | ✓ | | 96.69 \pm 2.24 | 93.82 \pm 3.34 | 42.61 \pm 4.80 | 17.53 \pm 5.18 | 10.09 \pm 3.54 |
| | ✓ | ✓ | | | | ✓ | 97.47 \pm 1.75 | 94.53 \pm 2.82 | 51.82 \pm 4.60 | 21.69 \pm 4.61 | 17.36 \pm 2.34 |

the sharp decline in Gold (5.47%) and Diamond (3.52%) indicates persistent challenges in sparse reward scenarios. Integrating the Reflection module (P+C+R) achieves near-perfect Wood/Stone success (96.69%/93.82%) and significantly boosts Iron (42.61%), Gold (17.53%), and Diamond (10.09%), demonstrating its role in distilling exploration experiences to refine world models.

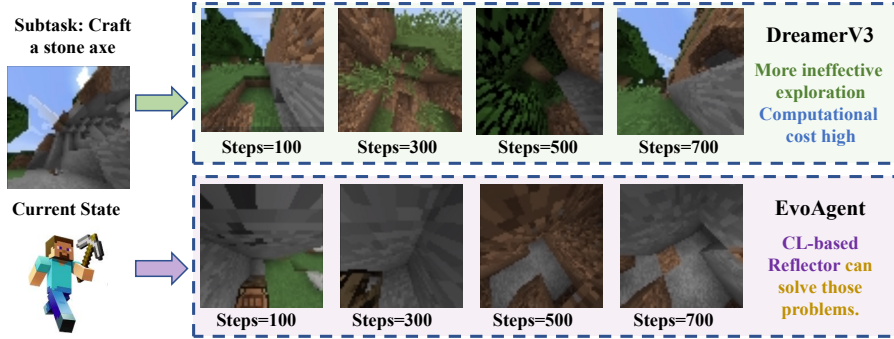


Figure 3: Illustration of the role of CL-based reflector.

This experiment compares the results of Planning with LoRA and Planning without LoRA, demonstrating that LoRA training can significantly improve model convergence speed, reduce the number of invalid subtasks, and achieve autonomous task decomposition. We also compare the effects of Reflection with stage 1 only, Reflection with stage 2 only, and Reflection with both stages. The experimental results show that combining task-based curriculum learning with data-based curriculum learning can significantly improve the model’s average task success rate. EvoAgent is an improvement over DreamerV3. As shown in Figure 3, EvoAgent is significantly better than DreamerV3 because the CL-based reflector can greatly filter invalid exploration and accelerate model convergence.

5 CONCLUSION

This paper presents EvoAgent, a self-evolving agent with a continual World Model, which can autonomously complete various LH tasks across environments through self-planning, self-control, and self-reflection, without human intervention. EvoAgent contains three modules: the memory-driven planner, the WM-guided action controller, and the experience-inspired reflector. Moreover, we develop a continual World Model for EvoAgent, which can autonomously update the multimodal experience pool and world knowledge through closed-loop dynamics. In the future, we hope that our method can be truly applied to real robot scenarios.

REFERENCES

- PKU BAAI. Plan4mc: Skill reinforcement learning and planning for open-world minecraft tasks. *arXiv preprint arXiv:2303.16563*, 2023.
- Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of artificial intelligence research*, 47: 253–279, 2013.
- Xiaoshuai Chen, Wei Chen, Dongmyoung Lee, Yukun Ge, Nicolás Rojas, and Petar Kormushev. A backbone for long-horizon robot task understanding. *IEEE Robotics and Automation Letters*, 10: 2048–2055, 2024.
- Rohan Chitnis, Yingchen Xu, Bobak Hashemi, Lucas Lehnert, Urun Dogan, Zheqing Zhu, and Olivier Delalleau. Iql-td-mpc: Implicit q-learning for hierarchical model predictive control. *arXiv preprint arXiv:2306.00867*, 2023.
- Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12873–12883, 2021.
- Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *Advances in Neural Information Processing Systems*, 35: 18343–18362, 2022.
- Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I Jordan, Joseph E Gonzalez, and Sergey Levine. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018.
- Weihang Guo, Zachary K. Kingston, and Lydia E. Kavraki. Castl: Constraints as specifications through llm translation for long-horizon task and motion planning. *ArXiv*, 2024.
- David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.
- Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019a.
- Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Learning latent dynamics for planning from pixels. *International Conference on Machine Learning (ICML)*, 2019b.
- Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, pp. 1–7, 2025.
- Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. *arXiv preprint arXiv:2310.16828*, 2023.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuezhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.
- Rajendrasinh Jadeja, Tapankumar Trivedi, and Jaymit Surve. Survivor detection approach for post earthquake search and rescue missions based on deep learning inspired algorithms. *Scientific Reports*, 14(1):25047, 2024.
- Thomas Kwa, Ben West, Joel Becker, Amy Deng, Katharyn Garcia, Max Hasin, Sami Jawhar, Megan Kinniment, Nate Rush, Sydney Von Arx, et al. Measuring ai ability to complete long tasks. *arXiv preprint arXiv:2503.14499*, 2025.
- Zaijing Li, Yuquan Xie, Rui Shao, Gongwei Chen, Dongmei Jiang, and Liqiang Nie. Optimus-1: Hybrid multimodal memory empowered agents excel in long-horizon tasks, 2024.
- Zaijing Li, Yuquan Xie, Rui Shao, Gongwei Chen, Dongmei Jiang, and Liqiang Nie. Optimus-2: Multimodal minecraft agent with goal-observation-action conditioned policy. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pp. 9039–9049, June 2025.

- Siyuan Liu, Jiawei Du, Sicheng Xiang, Zibo Wang, and Dingsheng Luo. Relep: A novel framework for real-world long-horizon embodied planning. *ArXiv*, 2024.
- Hao Luo, Jianjun Wei, Shuchen Zhao, Ankai Liang, Zhongjin Xu, and Ruxue Jiang. Enhancing robot route optimization in smart logistics with transformer and gnn integration. *arXiv preprint arXiv:2501.02749*, 2025.
- Michel Ma, Tianwei Ni, Clement Gehring, Pierluca D’Oro, and Pierre-Luc Bacon. Do transformer world models give better policy gradients? *International Conference on Machine Learning (ICML)*, 2024.
- Hani J Marcus, Pedro T Ramirez, Danyal Z Khan, Hugo Layard Horsfall, John G Hanrahan, Simon C Williams, David J Beard, Rani Bhat, Ken Catchpole, Andrew Cook, et al. The ideal framework for surgical robotics: development, comparative evaluation and long-term monitoring. *Nature medicine*, 30(1):61–75, 2024.
- Paul Mattes, Rainer Schlosser, and Ralf Herbrich. Hieros: Hierarchical imagination on structured state space sequence world models. *International Conference on Machine Learning (ICML)*, 2023.
- Pietro Mazzaglia, Tim Verbelen, B. Dhoedt, Aaron C. Courville, and Sai Rajeswar. Multimodal foundation world models for generalist embodied agents. *ArXiv*, 2024a.
- Pietro Mazzaglia, Tim Verbelen, Bart Dhoedt, Aaron C. Courville, and Sai Rajeswar. Genrl: Multimodal-foundation world models for generalization in embodied agents. In *The Thirty-Eighth Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024b.
- Yao Mu, Qinglong Zhang, Mengkang Hu, Wenhai Wang, Mingyu Ding, Jun Jin, Bin Wang, Jifeng Dai, Yu Qiao, and Ping Luo. Embodiedgpt: vision-language pre-training via embodied chain of thought. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NeurIPS)*, 2023.
- Siddharth Nayak, Adelmo Morrison Orozco, Marina Ten Have, Vittal Thirumalai, Jackson Zhang, Darren Chen, Aditya Kapoor, Eric Robinson, Karthik Gopalakrishnan, James Harrison, Brian Ichter, Anuj Mahajan, and Hamsa Balakrishnan. LLaMAR: Long-Horizon Planning for Multi-Agent Robots in Partially Observable Environments. *International Conference on Machine Learning (ICML)*, 2025.
- Pengzhen Ren, Kaidong Zhang, Hetao Zheng, Zixuan Li, Yuhang Wen, Fengda Zhu, Shikui Ma, and Xiaodan Liang. Surfer: A world model-based framework for vision-language robot manipulation. *IEEE Transactions on Neural Networks and Learning Systems*, 2025.
- Alexis Roger, Prateek Humane, Daniel Z Kaplan, Kshitij Gupta, Qi Sun, George Adamopoulos, Jonathan Siu Chi Lim, Quentin Anthony, Edwin Fennell, and Irina Rish. Robin: a suite of multi-scale vision-language models and the chirp evaluation benchmark. In *ICLR*, 2025.
- Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.
- Yutong Shen, Hangxu Liu, Penghui Liu, Ruizhe Xia, Tianyi Yao, Yitong Sun, and Tongtong Feng. Detach: Cross-domain learning for long-horizon tasks via mixture of disentangled experts. *arXiv preprint arXiv:2508.07842*, 2025.
- Xinshuai Song, Weixing Chen, Yang Liu, Weikai Chen, Guanbin Li, and Liang Lin. Towards long-horizon vision-language navigation: Platform, benchmark and method. *NeurIPS*, 2024.
- Ruixiang Sun, Hongyu Zang, Xin Li, and Riashat Islam. Learning latent dynamic robust representations for world models. *International Conference on Machine Learning (ICML)*, 2024.

- Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023a.
- Yucen Wang, Shenghua Wan, Le Gan, Shuai Feng, and De-Chuan Zhan. Ad3: Implicit action is the key for world models to distinguish the diverse visual distractors. *International Conference on Machine Learning (ICML)*, 2024.
- Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. *arXiv preprint arXiv:2302.01560*, 2023b.
- Zihao Wang, Shaofei Cai, Anji Liu, Yonggang Jin, Jinbing Hou, Bowei Zhang, Haowei Lin, Zhaofeng He, Zilong Zheng, Yaodong Yang, Xiaojian Ma, and Yitao Liang. Jarvis-1: Open-world multi-task agents with memory-augmented multimodal language models. *ArXiv*, 2023c.
- Zihao Wang, Shaofei Cai, Anji Liu, Yonggang Jin, Jinbing Hou, Bowei Zhang, Haowei Lin, Zhaofeng He, Zilong Zheng, Yaodong Yang, Xiaojian Ma, and Yitao Liang. Jarvis: Open-world multi-task agents with memory-augmented multimodal language models, 2023d.
- Zhutian Yang, Caelan Reed Garrett, Dieter Fox, Tom’as Lozano-P’erez, and Leslie Pack Kaelbling. Guiding long-horizon task and motion planning with vision language models. *ICLR*, 2024.
- Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan-Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, and Xipeng Qiu. Vlabench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. In *ICML*, 2024.
- Sipeng Zheng, Jiazheng Liu, Yicheng Feng, and Zongqing Lu. Steve-eye: Equipping llm-based embodied agents with visual perception in open worlds. *arXiv preprint arXiv:2310.13255*, 2023.
- Siyuan Zhou, Yilun Du, Jiaben Chen, Yandong Li, D. Y. Yeung, and Chuang Gan. Robodreamer: Learning compositional world models for robot imagination. *International Conference on Machine Learning (ICML)*, 2024.

A CONTINUAL WORLD MODEL

Algorithm 1 Continual World Model via Closed-Loop Planning-Control-Reflection

Require: Environment \mathcal{E} , Task \mathcal{T} , initial MEP $\mathcal{D}_{\text{MEP}}^0$ and \mathcal{M}_w^0 , Horizon H , Max steps T_{max}

Ensure: Optimized $\mathcal{D}_{\text{MEP}}^*$, \mathcal{M}_w^*

```

1: Current state  $\mathcal{S} \leftarrow (\mathcal{O}_{\text{obs}}, \mathcal{S}_{\text{self}}, \mathcal{S}_{\text{assets}})$ 
2: for Task  $\mathcal{T} = \mathcal{T}_0$  to  $\mathcal{T}_n$  do
3:    $\{g_i\} \leftarrow \Psi_{\text{plan}}(\mathcal{S}, \mathcal{T}, \mathcal{D}_{\text{MEP}})$  {Experience-driven task planner via Eq.(10-12)}
4:   for each subtask  $g_i \in \{g_i\}$  do
5:     for episode  $t = 1$  to  $T_{\text{max}}$  do
6:        $\{a_{t:t+H}\} \leftarrow \Pi_{\text{act}}(s_t, g_i, \mathcal{M}_w)$  {WM-guided action controller via Eq.(13-14)}
7:       if  $\phi_{\text{verify}}(s_t, g_i, t) = \text{Terminal}$  then
8:          $\mathcal{D}_{\text{MEP}} \leftarrow \mathcal{D}_{\text{MEP}} \cup \{(s_t, a_t, r_t, s_{t+1}, \mathbb{P}_{(g_i)}|g_i)\}$  {MEP updating via Eq.(15-16)}
9:         BREAK
10:      end if
11:    end for
12:     $\mathcal{D}_k^{\text{subtask}} \leftarrow \text{Curriculum\_Subtask\_Select}(\mathcal{G}_t, \mathcal{T}, \mathcal{D}_{\text{MEP}})$  {CL-based reflector via Eq.(17)}
13:     $\mathcal{D}_k^{\text{exp}} \leftarrow \text{Curriculum\_Experience\_Select}(\mathcal{D}_k^{\text{subtask}})$  {Two-stage CL via Eq.(18-21)}
14:     $\mathcal{M}_w' \leftarrow \Phi_{\text{reflect}}(\mathcal{D}_k^{\text{exp}}, \mathcal{M}_w)$  {WM updating via Eq.(22-23)}
15:  end for
16: end for

```

B ONLINE MODEL-BASED REINFORCEMENT LEARNING (MBRL)

RL is typically formulated as a Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where \mathcal{S} is the state space, \mathcal{A} is the action space, $P(s'|s, a)$ is the transition dynamics, $R(s, a)$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. The goal is to learn a policy $\pi(a|s)$ that maximizes the expected cumulative reward:

$$J(\pi) = \mathbb{E}_{\pi, P} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right], \quad (24)$$

In MBRL, the agent explicitly learns a model \mathcal{M} , which includes an approximate dynamics model $\hat{P}_{\theta}(s'|s, a)$ and a reward model $\hat{R}_{\phi}(s, a)$, parameterized by θ and ϕ , respectively. These models are trained to minimize empirical prediction errors over observed transitions $\mathcal{D} = \{(s_i, a_i, s'_i, r_i)\}$:

$$\mathcal{L}_{\text{model}}(\theta, \phi) = \mathbb{E}_{(s, a, s', r) \sim \mathcal{D}} \left[\|s' - \hat{P}_{\theta}(s, a)\|^2 + \|r - \hat{R}_{\phi}(s, a)\|^2 \right], \quad (25)$$

Using the learned models, the agent performs planning to optimize its policy. For example, in value iteration, the state-value function $V(s)$ is iteratively updated via the Bellman equation:

$$V(s) \leftarrow \max_a [\hat{R}_{\phi}(s, a) + \gamma \mathbb{E}_{s' \sim \hat{P}_{\theta}(\cdot|s, a)} V(s')]. \quad (26)$$

In online MBRL, an agent interacts with the environment iteratively for K rounds with the goal of learning a sequence to minimize $\mathcal{L}_{\text{model}}(\theta, \phi)$.

C HYPERPARAMETERS

Table 3: EvoAgent hyperparameters.

| General | | |
|---------------------------------------|-------------------------|----------------------------------------|
| Replay capacity | — | 5×10^6 |
| Batch size | B | 16 |
| Batch length | T | 64 |
| Activation | — | RMSNorm + SiLU |
| Learning rate | — | 4×10^{-5} |
| Gradient clipping | — | AGC(0.3) |
| Optimizer | — | LaProp($\epsilon = 10^{-20}$) |
| World Model | | |
| Reconstruction loss scale | β_{pred} | 1 |
| Dynamics loss scale | β_{dyn} | 1 |
| Representation loss scale | β_{rep} | 0.1 |
| Latent unimix | — | 1% |
| Free nats | — | 1 |
| Actor Critic | | |
| Imagination horizon | H | 15 |
| Discount horizon | $1/(1 - \gamma)$ | 333 |
| Return lambda | λ | 0.95 |
| Critic loss scale | β_{val} | 1 |
| Critic replay loss scale | β_{repval} | 0.3 |
| Critic EMA regularizer | — | 1 |
| Critic EMA decay | — | 0.98 |
| Actor loss scale | β_{pol} | 1 |
| Actor entropy regularizer | η | 3×10^{-4} |
| Actor unimix | — | 1% |
| Actor RetNorm scale | S | $\text{Per}(R, 95) - \text{Per}(R, 5)$ |
| Actor RetNorm limit | L | 1 |
| Actor RetNorm decay | — | 0.99 |
| WM-Guided Action Controller | | |
| Maximum episode step length | T_{max} | 24000 |
| Task similarity threshold | σ | 0.9 |
| Reward discount factor | γ | 0.1 |
| CL-based Reflector | | |
| CL algorithm initialization threshold | ρ_0 | 5×10^{-3} |
| CL subtask selection increase rate | c_s | 0.3 |
| CL experience selection increase rate | c_h | 0.5 |
| World model penalize weight | μ | 0.1 |

D GENERALIZATION EXPERIMENT

Atari100k. The Atari100k benchmark is widely regarded as a key platform for testing data-efficient reinforcement learning methods. Unlike typical setups that permit agents to interact with environments for hundreds of millions of steps, Atari100k imposes a strict cap of 100k interactions (maximum episode length is 432K env steps), equivalent to about two hours of human play. This limited interaction budget forces algorithms to develop effective policies rapidly, rather than depending on massive-scale exploration or brute-force training. The benchmark spans 26 distinct games from the Arcade Learning Environment, encompassing challenges such as sparse reward signals, delayed credit assignment, and high-dimensional pixel inputs. Results are usually reported using normalized human scores, ensuring comparability across games with diverse dynamics. By constraining available data so severely, Atari100k serves as a rigorous probe into the adaptability of reinforcement learning systems, offering insights into the generalization capacity of model-based approaches, world modeling strategies, and representation learning techniques.

Baselines. Random indicates that each action decision is randomly selected. Human refers to collecting video recordings of humans playing the game and calculating the average score. PPO Schulman et al. (2017) is a classic model-free reinforcement learning algorithm, and dreamerV3 Hafner et al. (2025) is a classic model-based reinforcement learning algorithm.

Experimental Settings. We adopt the same experimental settings as DreamerV3. Except for the EvoAgent experimental results, the rest are the publicly available experimental results of DreamerV3. The experimental results are shown in the Table 4.

Table 4: Atari100k scores.

| Task | Random | Human | PPO | DreamerV3 | EvoAgent |
|-----------------|--------|-------|-------|--------------|--------------|
| Steps | — | — | 400K | 400K | 400K |
| Alien | 228 | 7128 | 276 | 1118 | 1392 |
| Amidar | 6 | 1720 | 26 | 97 | 329 |
| Assault | 222 | 742 | 327 | 683 | 981 |
| Asterix | 210 | 8503 | 292 | 1062 | 1492 |
| Bank Heist | 14 | 753 | 14 | 398 | 362 |
| Battle Zone | 2360 | 37188 | 2233 | 20300 | 24830 |
| Boxing | 0 | 12 | 3 | 82 | 91 |
| Breakout | 2 | 30 | 3 | 10 | 13 |
| Chopper Command | 811 | 7388 | 1005 | 2222 | 4375 |
| Crazy Climber | 10780 | 35829 | 14675 | 86225 | 78215 |
| Demon Attack | 152 | 1971 | 160 | 577 | 1205 |
| Freeway | 0 | 30 | 2 | 0 | 5 |
| Frostbite | 65 | 4335 | 127 | 3377 | 3674 |
| Gopher | 258 | 2412 | 368 | 2160 | 2219 |
| Hero | 1027 | 30826 | 2596 | 13354 | 12168 |
| Jamesbond | 29 | 303 | 41 | 540 | 621 |
| Kangaroo | 52 | 3035 | 55 | 2643 | 2753 |
| Krull | 1598 | 2666 | 3222 | 8171 | 10027 |
| Kung Fu Master | 258 | 22736 | 2090 | 25900 | 28692 |
| Ms Pacman | 307 | 6952 | 366 | 1521 | 3246 |
| Pong | -21 | 15 | -20 | -4 | -2 |
| Private Eye | 15 | 69571 | 100 | 3238 | 5285 |
| Qbert | 164 | 13455 | 317 | 2921 | 4793 |
| Road Runner | 12 | 7845 | 602 | 19230 | 21703 |
| Seaquest | 68 | 42055 | 305 | 962 | 2305 |
| Up N Down | 533 | 11693 | 1502 | 46910 | 37284 |

Experimental results analysis. As shown in Table 4, the experimental results highlight that EvoAgent achieves superior generalization across the Atari100k suite, outperforming existing methods and achieving human-level performance in some tasks. This advantage arises from the continual world model combined with its iterative planning–control–reflection cycle, enabling the agent to avoid narrow overfitting and adapt to novel task dynamics.