Optimus-2 🤖 : Multimodal Minecraft Agent with Goal-Observation-Action Conditioned Policy

Zaijing Li[1][2], Yuquan Xie[1], Rui Shao[1]*, Gongwei Chen[1], Dongmei Jiang[2], Liqiang Nie[1]*

[1]Harbin Institute of Technology, Shenzhen   [2]Peng Cheng Laboratory

{lzj14011,xieyuquan20016}@gmail.com, {shaorui,nieliqiang}@hit.edu.cn

https://cybertronagent.github.io/Optimus-2.github.io/

Abstract

*Building an agent that can mimic human behavior patterns to accomplish various open-world tasks is a long-term goal. To enable agents to effectively learn behavioral patterns across diverse tasks, a key challenge lies in modeling the intricate relationships among observations, actions, and language. To this end, we propose Optimus-2, a novel Minecraft agent that incorporates a Multimodal Large Language Model (MLLM) for high-level planning, alongside a **G**oal-**O**bservation-**A**ction Conditioned **P**olicy (**GOAP**) for low-level control. GOAP contains (1) an Action-guided Behavior Encoder that models causal relationships between observations and actions at each timestep, then dynamically interacts with the historical observation-action sequence, consolidating it into fixed-length behavior tokens, and (2) an MLLM that aligns behavior tokens with open-ended language instructions to predict actions auto-regressively. Moreover, we introduce a high-quality **M**inecraft **G**oal-**O**bservation-Action (**MGOA**) dataset, which contains 25,000 videos across 8 atomic tasks, providing about 30M goal-observation-action pairs. The automated construction method, along with the MGOA dataset, can contribute to the community's efforts to train Minecraft agents. Extensive experimental results demonstrate that Optimus-2 exhibits superior performance across atomic tasks, long-horizon tasks, and open-ended instruction tasks in Minecraft. Please see the project page at https://cybertronagent.github.io/Optimus-2.github.io/.*

1. Introduction

Enabling agents to learn human behavioral patterns for completing complex tasks in open-world environments, is a long-standing goal in the field of artificial intelligence [5, 23, 34, 47]. To effectively handle diverse tasks in an
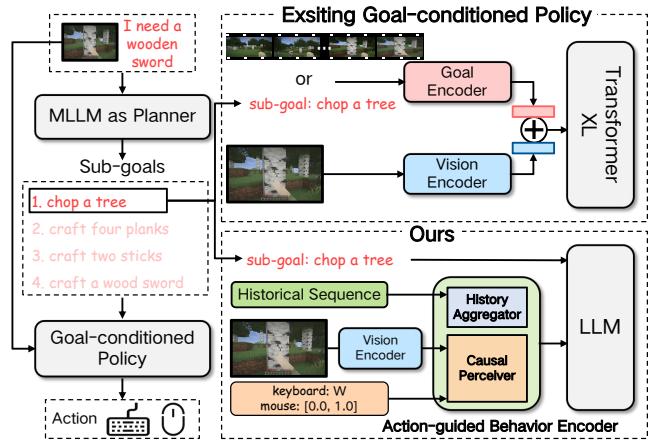
Figure 1. **Left**: General agent framework. **Right**: Comparison between existing goal-conditioned policies and ours. Existing Transformer-XL-based policies [3, 25] exhibit limited natural language understanding capabilities and rely solely on combining implicit goal embeddings with visual embeddings as inputs. In contrast, our GOAP achieves superior action prediction by 1) employing an Action-guided behavior encoder to strengthen causal modeling between observations and actions, as well as to improve historical sequence modeling capabilities, and 2) leveraging MLLM to enhance open-ended language comprehension.

open-world environment like Minecraft [20, 32], a prominent agent framework [24, 32, 41, 42] integrates a task planner with a goal-conditioned policy. As illustrated in Figure 1 (left), this framework first utilizes the task planner's language comprehension and visual perception abilities to decompose complex task instructions into sequential sub-goals. These sub-goals are then processed by a goal-conditioned policy to generate actions.

Although existing agents [24, 32, 42] have made promising progress by using Multimodal Large Language Models (MLLM) [4, 37, 45] as planners, the current performance bottleneck for agents lies in the improvement of the goal-

conditioned policy [24]. As the sub-goal serves as a natural language description of an observation-action sequence, the goal-conditioned policy needs to learn the crucial relationships among sub-goals, observations, and actions to predict actions. However, existing goal-conditioned policies exhibit the following limitations: **(1)** Existing policies neglect the modeling of the relationship between observations and actions. As shown in Figure 1, they only model the relationship between the sub-goal and the current observation by adding the sub-goal embedding to the observation features [3, 25, 43]. However, the current observation is generated by the previous action interacting with the environment. This implies a causal relationship between action and observation, which is neglected by current policies; **(2)** Existing policies struggle to model the relationship between open-ended sub-goals and observation-action sequences. As depicted in Figure 1, existing policies primarily rely on either video encoders [3, 43] or conditional variational autoencoders (CVAE) [25] as goal encoder to produce implicit goal embeddings. Such embeddings have limited representation ability [43]. Simply adding it to observation features is sub-optimal and unable to handle the complex relationship between sub-goals and observation-action sequences.

In this paper, we propose **Optimus-2**, a novel agent that incorporates an MLLM for planning, alongside a **G**oal-**O**bservation-**A**ction Conditioned **P**olicy (GOAP). To address the aforementioned challenges, we propose GOAP, which can better model the relationship among the observations, actions, and sub-goals in two aspects.

An Action-guided Behavior Encoder for observation-action sequence modeling. To capture the relationship between observations and actions, the Action-guided Behavior Encoder first employs a Causal Perceiver to integrate action embeddings into observation features. It utilizes task-relevant action information as guidance to adjust the observation features, thereby providing fine-grained observation-action information for action prediction. Additionally, to model a long-term observation-action sequence without exceeding input length limitations, a History Aggregator is introduced to dynamically integrate current observation-action information with the historical sequence into fixed-length behavior tokens. Behavior tokens can capture the long-term dependencies of the observation-action sequence with a fixed and appropriate length. It enables the agent to predict actions that align with the logic of the observation-action sequence, rather than making isolated action predictions based solely on the current observation.

An MLLM to model the relationship between sub-goal and observation-action sequence. To explicitly encode the semantics of sub-goals, we introduce an MLLM as the backbone of GOAP. It aligns the sub-goal with behavior tokens to predict subsequent actions auto-regressively.

Leveraging the MLLM's language comprehension and multimodal perception capabilities, it can better integrate features from open-ended sub-goals and observation-action sequences, thereby enhancing the policy's action prediction ability. To the best of our knowledge, GOAP is the first effort to employ MLLM as the core architecture of a Minecraft policy, which demonstrates strong instruction comprehension capabilities for open-ended sub-goals.

Moreover, current Minecraft datasets either lack alignment among essential elements [10] or are not publicly accessible [1], resulting in a significant scarcity of high-quality observation-goal-action pairs necessary for policy training. To this end, we introduce an automated approach for constructing the **Minecraft Goal-Observation-Action (MGOA)** dataset. The MGOA dataset comprises 25,000 videos across 8 atomic tasks, providing approximately 30 million aligned observation-goal-action pairs. It will be made openly available to support advancements within the research community. We conducted comprehensive evaluations in the open-world environment of Minecraft, and the experimental results demonstrate that Optimus-2 achieves superior performance. Compared to previous SOTA, Optimus-2 achieves an average improvements of 27%, 10%, and 18% on atomic tasks, long-horizon tasks, and open-ended sub-goal tasks, respectively.

In summary, our contributions are as follows:

- We propose a novel agent Optimus-2, which consists of an MLLM for planning, and a policy for low-level control. The experimental results demonstrate that Optimus-2 exhibits superior performance on atomic tasks, long-horizon tasks, and open-ended sub-goal tasks.
- To better model the relationship among the observations, actions, and sub-goals, we propose Goal-Observation-Action Conditioned Policy, GOAP. It contains an Action-guided Behavior Encoder for observation-action sequence modeling, and an MLLM to model the relationship between sub-goal and observation-action sequence.
- To address the scarcity of large-scale, high-quality datasets, we introduce the MGOA dataset. It comprises approximately 30 million aligned observation-goal-action pairs and is generated through an automated process without any manual annotations. The proposed dataset construction method and the released MGOA dataset can contribute to the community's efforts to train agents.

2. Related Work

Minecraft Agents. Previous works [2, 8, 13, 31] have constructed policies in Minecraft using reinforcement learning or imitation learning. VPT [1] was training on large-scale video data recorded by human players, using behavior cloning to mimic human behavior patterns. GROOT [3] employs a video encoder as a goal encoder to learn semantic information from videos. However, these policies rely solely
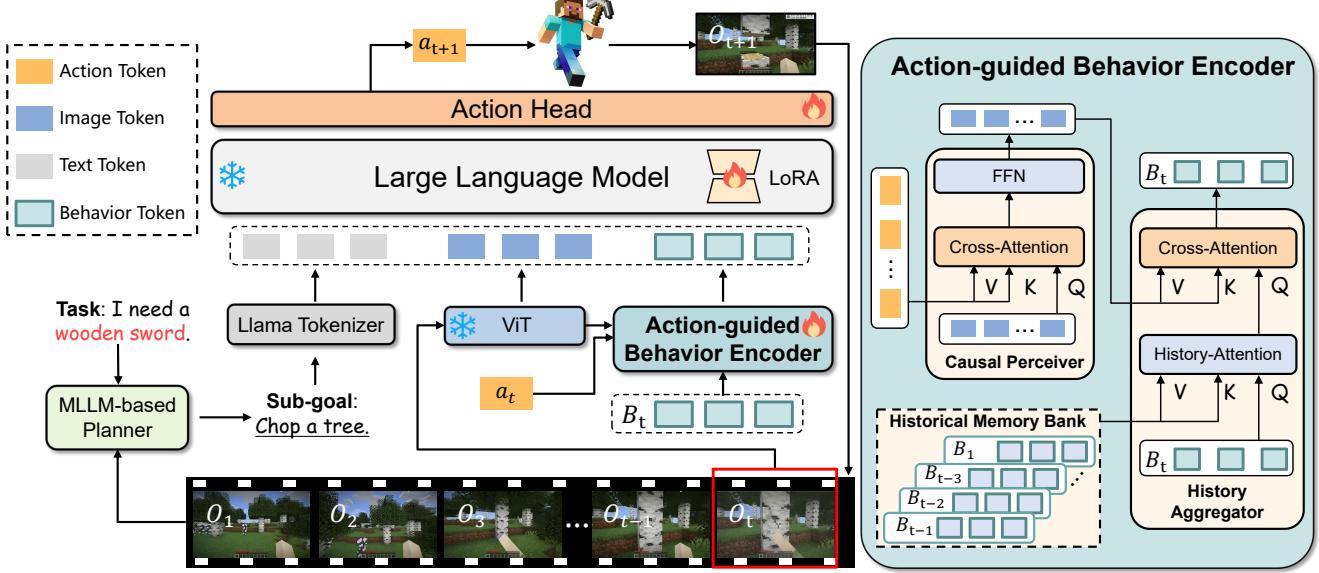
Figure 2. Overview of Optimus-2. Given a task and the current observation, Optimus-2 first uses an MLLM-based Planner to generate a series of sub-goals. Optimus-2 then sequentially executes these sub-goals through GOAP. GOAP obtains behavior tokens for the current timestep via the Action-guided Behavior Encoder, and these behavior tokens, along with image and text tokens, are fed into the LLM to predict subsequent actions.

on visual observations as input and cannot follow human instructions to accomplish specific tasks. MineCLIP [10] introduces a video-text contrastive learning module as a reward model for policy, and STEVE-1 [25] builds on VPT [1] by incorporating MineCLIP as goal encoder, enabling policy to follow natural language instructions. Despite these advancements, these policies are constrained by language understanding and reasoning capabilities. To address this, current agents [20, 24, 32, 40, 42, 43] leverage MLLM's instruction following capabilities to decompose complex tasks into executable sub-goal sequences, which are then fed into a goal-conditioned policy [3, 25] or formed as executable code [26, 28, 51, 52]. Despite significant progress, the performance of current policies remains constrained by their limited ability to understand sub-goals. In this paper, we aim to develop an MLLM-based goal-conditioned policy to enhance the policy's comprehension of open-ended sub-goals, thereby improving overall performance.

Long-term Video Modeling. Previous work [1, 3, 10, 25] have segmented videos into multiple clips for training to alleviate the challenges posed by long-sequence video inputs. However, this approach prevents the agent from learning comprehensive behavior representations from the entire video. To handle long-term video sequences [22, 48, 49], existing studies employ temporal pooling [30], querying transformers [14, 46], or token merging [16, 38, 50] to integrate long-sequence visual tokens. Inspired by previous works [6, 18, 19, 44], we propose a Q-former [7, 21]

structure with a memory bank [14], enabling effective long-term sequence modeling through interactions with historical queries. Unlike existing methods that model only the observation sequence, we focus on multimodal learning [33, 35, 36]. Moreover, different from previous work [14] that primarily compress video features into fixed-length tokens, our Action-guided Behavior Encoder dynamically interacts with the historical sequence at each timestep, producing behavior tokens corresponding to the observation-action sequence from the start to the current timestep.

3. Preliminaries and Problem Formulation

In Minecraft, agents [1, 3, 25] exhibit behavior patterns similar to humans: at each time step t, the agent receives a visual observation o_t and generates control actions a_{t+1} using the mouse and keyboard. These actions interact with the environment, resulting in a new visual observation o_{t+1}. Through continuous interactions, a trajectory $J = \{(o_1, a_1), (o_2, a_2), (o_3, a_3), \ldots, (o_T, a_T)\}$ is formed, where T represents the length of the trajectory. Previous work primarily trained Minecraft agents using reinforcement learning [10] or behavior cloning [3, 25]. For example, in behavior cloning, the goal of the policy $p_\theta(a_{t+1}|o_{1:t})$ is to minimize the negative log-likelihood of the actions at each time step t given the trajectory J. Considering that such trajectories are typically generated under explicit or implicit goals, many recent approaches condition the behavior on a (implicit or explicit) goal g and learn goal-

conditioned policy $p_\theta(a_{t+1}|o_{1:t}, g)$ [3, 25]. Generally, for both agents and humans, the explicit goal g is a natural language instruction.

Formally, given a trajectory J with length T, standard behavior cloning trains the policy $p_\theta(\cdot)$ with parameters θ by minimizing the negative log-likelihood of actions:

$$\min_\theta \sum_{t=1}^{T} -\log p_\theta(a_{t+1}|o_{1:t}, g) \quad (1)$$

4. Optimus-2

In this section, we first give an overview of our proposed agent framework, Optimus-2. As shown in Figure 1 (left), it includes a planner for generating a series of executable sub-goals and a policy that sequentially executes these sub-goals to complete the task.

Next, we introduce how to implement Optimus-2's planner (Sec. 4.1). Subsequently, we elaborate on how to implement the proposed GOAP (Sec. 4.2). Finally, in Sec 4.3, we introduce an automated dataset generation method to obtain a high-quality Minecraft Goal-Observation-Action dataset (MGOA) for training GOAP.

4.1. MLLM-based Task Planner

In Minecraft, a complex task consists of multiple intermediate steps, i.e., sub-goals. For example, the task "I need a wooden pickaxe" includes five sub-goals: 'chop a tree to get logs 🪵', 'craft four planks 🟫', 'craft a crafting table 🟫', 'craft two sticks ╱', and 'craft a wooden pickaxe ⛏'. Therefore, a planner is essential for the agent, as it needs to decompose the given complex task into a sequence of executable sub-goals for the policy to execute sequentially. In this paper, we follow Li et al. [24], employing an MLLM as the planner, which takes current observation and task instruction as input to generate sub-goals.

4.2. Goal-Observation-Action Conditioned Policy

According to Sec 3., a key insight into the relationship among observation o, action a, and sub-goal g is: that the observation o and action a at the same time step have a causal relationship; and the sub-goal g is a natural language description of the observation-action sequence over a certain time. To better model the relationships among the three elements mentioned above, we propose first integrating the representations of observation and action at each time step, then modeling the observation-action sequences along the temporal dimension, and finally aligning the observation-action sequences with the sub-goal for action prediction.

Motivated by this, we propose a novel Goal-Observation-Action conditioned Policy, GOAP. As shown in Figure 2, our GOAP consists of an Action-guided Behavior Encoder that dynamically models observation-action sequences into fixed-length behavior tokens and an

Table 1. Comparison of the MGOA dataset with existing datasets. O, G, and A represent observation, goal, and action. VPT[†] indicates the amount of data that is openly accessible. MineCLIP[‡] denotes narrated Minecraft videos available on YouTube.

Format	Dataset	O	G	A	# Frames
Image-Text Pairs	MP5 [32]	✓	✓		500K
	OmniJARVIS [43]	✓	✓	✓	600K
Gameplay Video	VPT[†] [1]	✓		✓	6M
	MineCLIP[‡] [10]	✓	✓		20B
	STEVE-1 [25]	✓	✓	✓	32K
	MGOA (Ours)	✓	✓	✓	30M

MLLM that aligns such behavior tokens with sub-goal for action prediction.

4.2.1. Action-guided Behavior Encoder

Previous policies often overlook the causal relationship between observation and action at each timestep. Moreover, it remains a challenge to model the long-term observation-action sequence without exceeding input length constraints. To this end, we propose an Action-guided Behavior Encoder that integrates the representations of observation and action at each time step and then dynamically models the historical sequences into the fix-length behavior tokens.

Firstly, for the timestep t, we pass observation o_t into a visual encoder $\mathtt{VE}$ to obtain the visual features:

$$v_t \leftarrow \mathtt{VE}(o_t) \quad (2)$$

where $v_t \in \mathbb{R}^{P \times d}$, P is the number of patches for each image, and d is the dimension of the extracted image feature. In practice, we employ ViT [9] as our visual encoder.

Then, we introduce a **Causal Perceiver** module to model the relationship between observations and actions. It takes the visual feature v_t as query tokens and the action embedding a_t as key and value. The module then constructs the information interaction between action a_t and v_t through a cross-attention mechanism:

$$Q = v_t W_v^Q, K = a_t W_a^K, V = a_t W_a^V \quad (3)$$

$$\hat{v}_t = CrossAttn(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d}})V \quad (4)$$

where W_v^Q, W_a^K, and W_a^V represent the weight matrices for the query (Q), key (K), and value (V), respectively. $CrossAttn(\cdot)$ denotes the cross-attention layer, and d is the dimension of the image features. In this way, it explicitly assigns action information a_t at time step t to the visual features $\hat{v}_t$, enhancing the causal relationship between observations and actions.

Subsequently, we introduce a **History Aggregator** module to capture the information of the observation-action sequence along the temporal dimension, serving as the behavior representation. At each timestep t, behavior tokens B_t

serve as queries, while the sequence of historical behavior tokens $H_t = [B_1, B_2, \ldots, B_{t-1}]$ acts as keys and values. The current behavior tokens interact with the historical sequence through a history-attention layer $HisAttn(\cdot)$:

$$\hat{B}_t = HisAttn(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d}})V \quad (5)$$

where Q, K, and V are calculated similarly to Eq 3.

Finally, another cross-attention layer is introduced, using the behavior tokens $\hat{B}_t$ as queries, and the visual features $\hat{v}_t$ as keys and values. In this way, the behavior tokens incorporate the current observation-action information. Following the approach of He et al. [14], we construct a memory bank for historical behavior tokens H_t, utilizing the similarity between adjacent features to aggregate and compress the behavior tokens. This method not only preserves early historical information but also keeps the historical behavior token sequence H_t at a fixed length to reduce computational costs. Leveraging the Action-guided Behavior Encoder, we obtain behavior tokens $\hat{B}_t$, which correspond to the observation-action sequence from the start to the current time step t.

4.2.2. MLLM Backbone

To model the relationship between the sub-goal and observation-action sequence, we introduce an MLLM that takes the sub-goal g, current observation features v_t, and behavior tokens B_t as input to predict subsequent actions auto-regressively. To enable the MLLM backbone `MLLM` to predict low-level actions, we employ VPT [1] as action head `AH` to map output embeddings $\bar{a}_{t+1}$ of language model into the action space.

$$\bar{a}_{t+1} \leftarrow \text{MLLM}([g, v_t, B_t]) \quad (6)$$

$$a_{t+1} \leftarrow \text{AH}(\bar{a}_{t+1}) \quad (7)$$

Formally, given a dataset $\mathcal{D} = \{(o_{1:T}, a_{1:T})\}_M$ with M complete trajectories, we train GOAP to learn the behavior distribution from $\mathcal{D}$ via behavioral cloning. Moreover, we introduce a KL-divergence loss to measure the output distribution similarity between GOAP and VPT [1]. This helps our model effectively learn the knowledge from the teacher model VPT. The training loss can be formulated as follows:

$$\mathcal{L}_\theta = \lambda_{BC} \sum_{t=1}^{T} -\log p_\theta(a_{t+1}|o_{1:t}, a_{1:t}, g)$$

$$+\lambda_{KL} \sum_{t=1}^{T} D_{KL}(q_\phi(a_{t+1}|o_{1:t}) \| p_\theta(a_{t+1}|o_{1:t}, g)) \quad (8)$$

where λ_{BC} and λ_{KL} are trade off coefficients, p_θ is the GOAP, q_ϕ is the teacher model.

Table 2. Main Result of GOAP on *Atomic Tasks*. We report the average rewards of each task.

Policy	Logs	Seeds	Dirt	Stone
VPT (text) [1]	2.6	0.8	9.2	0.0
STEVE-1 [25]	11.0	5.1	10.0	3.2
GROOT [3]	14.3	7.3	19.7	19.0
FSQ GROOT [43]	10.8	8.2	20.3	5.8
GOAP $_{[MLP]}$	7.2	4.3	14.4	15.5
GOAP $_{[VPT]}$	**15.0**	**8.5**	**26.7**	**25.7**

4.3. MGOA Dataset

In Minecraft, there remains a significant lack of high-quality goal-observation-action pairs to support behavior cloning training. Previous work has primarily relied on gameplay videos as training data. These datasets either lack natural language instructions (explicit goals) [1, 3], or use actions predicted by IDM models [1] for each observation as pseudo-labels [1, 25], which leads to a risk of misalignment between observations and actions. Inspired by Li et al. [24], we propose an automated data generation pipeline that enables the creation of aligned goal-observation-action pairs without the need for manual annotations or human contractors. First, we utilize existing agents [25], providing them with clear natural language instructions to attempt task completion in Minecraft. We then record the actions and corresponding observations during goal execution, generating goal-observation-action pairs.

To ensure the quality of the generated data, we apply the following filtering criteria: 1) only recording videos in which the task is successfully completed, and 2) discarding videos where task execution takes an excessive amount of time. For more details, please refer to **Sup. C**. Through this automated approach, we obtained 25k high-quality Minecraft Goal-Observation-Action (MGOA) dataset. A comparison of the MGOA dataset with the existing Minecraft datasets is shown in Table 1. Our automated data generation pipeline offers several key advantages: 1) it enables the generation of aligned goal-observation-action pairs without the need for manual annotation or pseudo-labeling; 2) its construction process is parallelizable, allowing for rapid dataset generation; and 3) it leverages local agents for data generation, resulting in low-cost production.

5. Experiments

5.1. Experiments Setting

Environment. Following [1, 25], we conduct experiments in the complex, open-world environment of Minecraft on the MineRL [12] platform. The agent interacts with the MineRL environment at 20 frames per second, generating low-level control signals for the mouse and keyboard. For each task execution, the agent is initialized in a randomized

Table 3. Main Result of Optimus-2 on *Long-horizon Tasks*. We report the average success rate (SR) on each task group, the results of each task can be found in the **Sup. F.1**. Pure GPT-4V[†] denotes the use of GPT-4V in a zero-shot manner to generate executable sub-goals for the policy. Human[‡] denotes the human-level baseline, with results sourced from previous work [24].

Method	Policy	Wood 🪵	Stone 🪨	Iron ⬜	Gold 🟨	Diamond 💎	RedStone 🟥	Armor 🛡
Pure GPT-4V [†]	VPT (text)	0.22	0.08	0.00	0.00	0.00	0.00	0.00
	STEVE-1	0.41	0.20	0.00	0.00	0.00	0.00	0.00
	GOAP	**0.50**	**0.31**	**0.12**	**0.02**	**0.01**	**0.03**	**0.03**
DEPS [41]	STEVE-1	0.77	0.48	0.16	0.00	0.01	0.00	0.10
Jarvis-1 [42]	STEVE-1	0.93	0.89	0.36	0.07	0.08	0.16	0.15
Optimus-1 [24]	STEVE-1	0.98	0.92	0.46	0.08	0.11	0.25	0.19
Optimus-2	GOAP	**0.99**	**0.93**	**0.53**	**0.09**	**0.13**	**0.28**	**0.21**
Human[‡] [24]	-	1.00	1.00	0.86	0.17	0.16	0.33	0.28

Table 4. Main Result of GOAP on *Open-Ended Instruction Tasks*. We report the average success rate (SR) on Torch 🕯, Rail 🛤, Golden Shovel 🔨, Diamond Pickaxe ⛏, and Compass 🧭. GROOT [3] and FSQ GROOT [43] were not included as baselines, as they are unable to process language input.

Planner	Policy	🕯	🛤	🔨	⛏	🧭
GLM-4V	VPT (text)	0.05	0	0	0	0
	STEVE-1	0.60	0	0	0	0
	GOAP	**0.71**	**0.39**	**0.11**	**0.14**	**0.13**
GPT-4V	VPT (text)	0.11	0	0	0	0
	STEVE-1	0.66	0.10	0	0	0
	GOAP	**0.75**	**0.47**	**0.13**	**0.16**	**0.17**

environment, allowing us to evaluate the agent's generalization across diverse environments. Please refer to **Sup. B** for more details about the Minecraft environment.

Implementation details. For the planner, we follow Li et al. [24], using a hybrid multimodal memory empowered GPT-4V [1] as the agent's planner. As for the policy, we initialize GOAP with the weights of DeepSeek-VL-1.3B [29] as initialization. We train it on the MGOA dataset and the publicly available OpenAI Contractor Dataset [1] through behavior cloning. All experiments were conducted on 8x NVIDIA L40 GPUs. Training details and hyperparameter setting can be found in **Sup. D**.

Evaluation Tasks & Metrics. Evaluation tasks are categorized into three types: *Atomic Tasks*, *Long-Horizon Tasks*, and *Open-Ended Instruction Tasks*. For each task, the environment is randomly reinitialized on each attempt, with a minimum of 30 executions per task to ensure robustness.

- *Atomic Tasks* represent short-term skills in Minecraft. We select "chop a tree to get logs 🪵", "collect seeds 🌱", "collect dirt 🟫", and "mine stone 🪨 with a pickaxe" as evaluation tasks. These tasks evaluate the policy's basic capabilities in Minecraft. We report the average rewards

(number of items obtained) per task execution as an evaluation metric.

- *Long-horizon Tasks* consist of an interdependent atomic tasks sequence, where the failure of any single atomic task results in the failure of the entire sequence. These long-horizon tasks are designed to evaluate the agent's capability to execute a series of diverse tasks continuously within a complex environment. We follow the setup of Li et al. [24], conducting experiments on long-horizon tasks comprising 67 tasks grouped into 7 categories. We report the average Success Rate (SR) as an evaluation metric.

- *Open-Ended Instruction Tasks* are not limited to predefined text formats; rather, they involve flexible language directives that prompt the agent to accomplish long-horizon tasks. These tasks evaluate the agent's capacity to interpret and execute instructions expressed in open-ended natural language. We selected the Torch 🕯 from the Stone Group, Rail 🛤 from the Iron Group, Golden Shovel 🔨 from the Gold Group, Diamond Pickaxe ⛏ from the Diamond Group, and Compass 🧭 from the Redstone Group as evaluation tasks. Given a crafting relationship graph, we instructed GPT-4V and GLM-4V [11] to generate five open-ended instructions for each task. This allows us to evaluate the policies' understanding and execution capabilities regarding open-ended instructions. Task instructions are provided in the **Sup. E.1**.

Baseline. For *Atomic Tasks* and *Open-ended Instruction Tasks*, we compare GOAP with existing goal-conditioned policies, including VPT [1], STEVE-1 [25], GROOT [3] and FSQ GROOT [43]. For *Long-horizon Tasks*, we employ GPT-4V, DEPS [41], Jarvis-1 [42], and Optimus-1 [24] as baselines. We also introduce a human-level baseline [24] to evaluate the performance gap between existing agents and human capabilities.

5.2. Experimental Results

The experimental results for Optimus-2 compared to the baselines across *Atomic Tasks*, *Long-horizon Tasks*, and

---
[1]https://openai.com/index/gpt-4v-system-card

Figure 3. An illustration of VPT (text) [1], STEVE-1 [25], and Optimus-2 executing the open-ended instruction, "I need some iron ores, what should I do?". Existing policies are limited by their instruction comprehension abilities and thus fail to complete the task, whereas GOAP leverages the language understanding capabilities of the MLLM, enabling it to accomplish the task.

Table 5. Ablation study of Action-guided Behavior Encoder on *Atomic Tasks*. We report average rewards on each task. `CP.`, `HA.`, and `MB.` represent the Causal Perceiver, History Aggregator, and Memory Bank, respectively.

Ablation Setting			Atomic Task				
`CP.`	`HA.`	`MB.`	Logs 🪵	Seeds 🌱	Dirt 🟫	Stone 🪨	Average
✓	✓	✓	**15.0**	**8.5**	**26.7**	**25.7**	**19.0**
✓			6.1	5.4	12.7	15.7	10.0 (↓ 47.4%)
	✓	✓	10.2	4.7	12.8	21.1	12.2 (↓ 35.8%)
✓	✓		7.4	6.2	13.1	15.5	10.6 (↓ 44.2%)
✓	✓		12.0	6.8	22.7	16.8	14.6 (↓ 23.2%)

Figure 4. Ablation of LLM backbone on *Open-ended Instruction Tasks*, Golden Shovel 🥄, Diamond Pickaxe ⛏, and Compass 🧭.

Open-ended Instruction Tasks are presented in Table 2, Table 3, and Table 4, respectively.

GOAP excels in Atomic Tasks. Table 2 shows that proposed GOAP achieves improvements of 5%, 4%, 31%, and 35% over the current SOTA on the Logs 🪵, Seeds 🌱, Dirt 🟫, and Stone 🪨, respectively. These results demonstrate that GOAP has successfully mastered a range of short-term skills across diverse environments, and can acquire items more effectively than existing policies.

Optimus-2 surpasses SOTA in Long-horizon Tasks. Table 3 shows that Optimus-2 achieved the highest success rates across all seven task groups, particularly excelling in the challenging Diamond Group and Redstone Group with success rates of 13% and 28%, respectively. This indicates that Optimus-2 has effectively learned complex behavior patterns across atomic tasks, enabling it to sequentially execute multiple sub-goals and successfully complete long-horizon tasks within complex environments.

GOAP outperforms in Open-ended Instruction Tasks. As shown in Table 4, GOAP achieved significantly higher success rates than existing agents across all tasks. Notably, on the challenging tasks of Golden Shovel 🥄, Diamond Pickaxe ⛏, and Compass 🧭, existing policies fail to complete these tasks, whereas GOAP achieves success rates of 13%, 16%, and 17%, respectively. This advantage stems from GOAP's superior comprehension of open-ended natural language instructions, whereas existing agents exhibit weaker instruction-following capabilities. Moreover, Figure 3 illustrates an example of different policies executing an open-ended goal. Due to the limited representation capability of their goal encoders, VPT [1] and STEVE-1 [25] fail to understand the goal, "I need some iron ores, what should I do?" In contrast, GOAP leverages the MLLM's understanding of open-ended instructions to effectively accomplish the goal (obtaining iron ore 🪨).

5.3. Ablation Study

There are many unexplored questions around best practices for developing MLLM-based policy in Minecraft. In this section, we conduct an extensive ablation study and summarize our key findings.

The Action-guided Behavior Encoder plays a crucial role in task execution. As shown in Table 5, the removal of the Causal Perceiver leads to an average performance decline of 42% across all tasks, highlighting the importance of capturing the causal relationship between observations and actions. Moreover, eliminating the History Aggregator
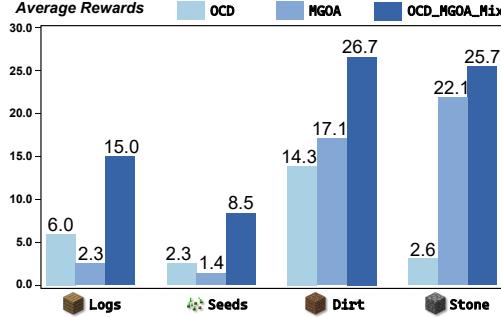
Figure 5. Ablation study on Training data. OCD refers to the OpenAI Contractor Dataset [1]. We report the average rewards on each *Atomic Task*.

and Memory Bank also results in an average performance decline of 36% across all tasks. This emphasizes the crucial role of the History Aggregator in modeling observation-action sequences and the Memory Bank in dynamically storing long-sequence information.

LLM significantly enhances policy's ability to understand open-ended instructions. As shown in Figure 4, replacing the LLM backbone with a Transformer-XL leads to a noticeable decline in performance. We attribute this to the LLM's pretraining on large-scale textual corpora, which endows it with a robust comprehension of open-ended language, a capability that Transformer-XL lacks.

A pretrained action head improves performance in Minecraft. As shown in Table 2, replacing VPT with a 2-layer MLP projector as the action head leads to a noticeable decline in Optimus-2's performance. While MLP-based action heads have shown promising results in other domains [17, 27], this substitution is less effective in the Minecraft environment. We attribute this to VPT's extensive pretraining on large-scale gameplay data, which equips it with substantial domain-specific knowledge critical for effective task execution in Minecraft.

The MGOA datsaset is beneficial for training GOAP. We conducted comparative experiments to evaluate the impact of different training datasets on performance. As shown in Figure 5, training only with the current most commonly used dataset, OpenAI Contractor Dataset (OCD), results in suboptimal performance for GOAP on all *Atomic Tasks*. For example, compared to training with a mixed dataset, its performance on Stone 🪨 dropped by 89%. We attribute this to the fact that OCD offers a wide variety of tasks but lacks high data quality. In contrast, using our MGOA dataset, performance on the four atomic tasks improved by an average of 70% compared to using only the OCD data. We attribute this to the fact that MGOA contains high-quality aligned goal-observation-action pairs, which is beneficial for policy training. Further, we mix the two datasets to train the policy in order to balance task diversity
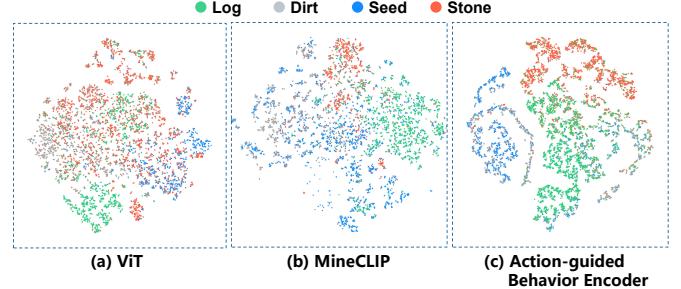

Figure 6. t-SNE visualization of representations extracted by (a) ViT (b) MineCLIP and (c) Action-guided Behavior Encoder across *Atomic Tasks*. The visualization results show that the representations in (a) and (b) cannot distinguish between different tasks, whereas our Action-guided Behavior Encoder clearly differentiates the behavior representations for the four tasks.

and data quality, leading to improved performance.

5.4. Visualization of Behavior Representation

As shown in Figure 6, we apply t-SNE [39] to visualize observation features extracted by ViT [9], MineCLIP [10], and the Action-guided Behavior Encoder for four tasks. From (a) and (b) in Figure 6, it is evident that the behavior representations extracted by ViT and MineCLIP are highly mixed, making it challenging to delineate the boundaries between different tasks. This lack of clear distinction between task-specific behavior representations can hinder the model's ability to understand the unique behavior patterns associated with each task, potentially leading to task failure. In contrast, the visualization in (c) of Figure 6 reveals clear, distinct clusters for each task, demonstrating that the Action-guided Behavior Encoder effectively captures subtle differences in observation-action sequences, thereby learning robust behavior representations across tasks.

6. Conclusion

In this paper, we propose a novel agent, Optimus-2, which can excel in various tasks in the open-world environment of Minecraft. Optimus-2 integrates an MLLM for high-level planning and a Goal-Observation-Action conditioned Policy (GOAP) for low-level control. As a core contribution of this paper, GOAP includes an Action-guided Behavior Encoder to model the observation-action sequence and an MLLM to align the goal with the observation-action sequence for predicting subsequent actions. Extensive experimental results demonstrate that GOAP has mastered various atomic tasks and can comprehend open-ended language instructions. This enables Optimus-2 to achieve superior performance on long-horizon tasks, surpassing existing SOTA. Moreover, we introduce a Minecraft Goal-Observation-Action dataset to provide the community with large-scale, high-quality data for training Minecraft agents.

Optimus-2 🤖 : Multimodal Minecraft Agent with Goal-Observation-Action Conditioned Policy

Supplementary Material

The supplementary document is organized as follows:

A. Limitation and Future Work

In this paper, we aim to explore how agents can mimic human behavior patterns in Minecraft to accomplish various tasks. Experimental results demonstrate that Optimus-2 performs exceptionally well in both atomic tasks and long-horizon tasks. However, due to the lack of sufficient high-quality data for open-ended tasks (such as "building a house" and "defeating the Ender Dragon"), there remains significant room for improvement. Once such datasets are available, the ability of Optimus-2 to complete open-ended tasks will be enhanced. Moreover, despite showing promising performance in Minecraft, we have not yet extended our exploration to other simulation platforms, which represents a potential direction for future research.

B. Minecraft

Minecraft is an extremely popular sandbox video game developed by Mojang Studios [2]. It allows players to explore a blockly, procedurally generated 3D world with infinite terrain, discover and extract raw materials, craft tools and items, and build structures or earthworks. In this environment, AI agents need to face situations that are highly similar to the real world, making judgments and decisions to deal with various environments and problems. As shown in Figure 7, both agents and humans are required to receive natural language instructions and current observations as input, and then output low-level actions, such as mouse and keyboard control commands. Therefore, Minecraft serves
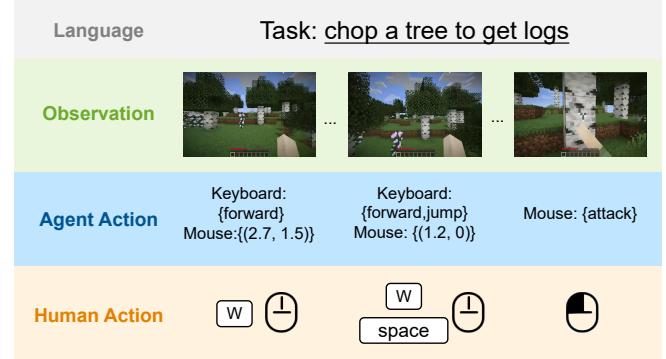

Figure 7. Illustration of behavior patterns of both human and agents in Minecraft.

as an ideal open-world environment for training agent that can learn human behavior patterns.

B.1. Basic Rules

Biomes. The Minecraft world is divided into different areas called "biomes". Different biomes contain different blocks and plants and change how the land is shaped. There are 79 biomes in Minecraft 1.16.5, including ocean, plains, forest, desert, etc. Diverse environments have high requirements for the generalization of agents.

Item. In Minecraft 1.16.5, there are 975 items can be obtained, such as wooden pickaxe ⛏, iron sword 🗡. Item can be obtained by crafting or destroying blocks or attacking entities. For example, agent can attack cows 🐄 to obtain leather 🟫 and beef 🥩. Agent also can use 1 stick 🪵 and 2 diamonds 💎 to craft diamond sword 🗡.

Technology Tree. In Minecraft, the technology hierarchy comprises six levels: wood 🟫, stone ⬛, iron ⬜, gold 🟨, diamond 💎, and redstone 🟥. Each tool level corresponds to specific mining capabilities. Wooden tools can mine stone-level blocks but are incapable of mining iron-level or higher-level blocks. Stone tools can mine iron-level blocks but cannot mine diamond-level or higher-level blocks. Iron tools are capable of mining diamond-level blocks. Finally, diamond tools can mine blocks of any level, including redstone-level.

Gameplay progress. Progression in Minecraft primarily involves discovering and utilizing various materials and resources, each unlocking new capabilities and opportunities. For instance, crafting a wooden pickaxe ⛏ enables players

Table 6. Action space of agent in Minecraft.

Index	Agent Action	Human Action	Description
1	Forward	key W	Move forward.
2	Back	key S	Move back.
3	Left	key A	Strafe left.
4	Right	key D	Strafe right.
5	Jump	key Space	Jump. When swimming, keeps the player afloat.
6	Sneak	key left Shift	Slowly move in the current direction of movement.
7	Sprint	key left Ctrl	Move quickly in the direction of current movement.
8	Attack	left Button	Destroy blocks (hold down); Attack entity (click once).
9	Use	right Button	Place blocks, entity, open items or other interact actions defined by game.
10	hotbar [1-9]	keys 1-9	Selects the appropriate hotbar item.
11	Open/Close Inventory	key E	Opens the Inventory. Close any open GUI.
12	Yaw	move Mouse X	Turning; aiming; camera movement.Ranging from -180 to +180.
13	Pitch	move Mouse Y	Turning; aiming; camera movement.Ranging from -180 to +180.
14	Craft	-	Execute a crafting recipe to obtain new item
15	Smelt	-	Execute a smelting recipe to obtain new item.

to mine stone ⬢, which can then be used to create a stone pickaxe ⬈ and a furnace ⬢. These tools allow for the mining and smelting of iron ore ⬢. Subsequently, crafting an iron pickaxe ⬈ enables the extraction of diamonds ◯, while a diamond pickaxe ⬈ can mine virtually any block in the game. Similarly, cultivating crops facilitates breeding various animals, each providing unique resources beyond sustenance. Drops from enemies also serve specific purposes, with some offering greater utility than others. By integrating resources from mining, farming, and breeding, players can enchant their equipment, further enhancing their capabilities. Additionally, collecting and crafting materials support construction, enabling players to create diverse structures. Beyond practical functions, such as building secure bases or farms, constructing personalized structures forms a significant aspect of the Minecraft experience. Figure 11 illustrates an example of progression: crafting an iron sword ⚔.

B.2. Observation and Action Spaces

Observation. In this paper, observation space of agent is completely consistent with human players. The agent only receives an RGB image with dimensions of 640×360 during the gameplay process, including the hotbar, health indicators, food saturation, and animations of the player's hands. It is worth helping the agent see more clearly in extremely dark environments, we have added a night vision effect for the agent, which increases the brightness of the environment during the night.

Action Spaces. In MineRL [12] environment, agent's action space is almost similar to human players. It consists of two parts: the mouse and the keyboard. The keypresses are
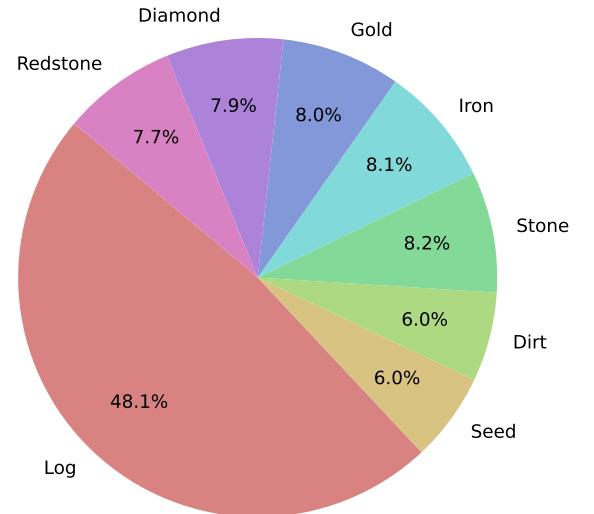
Figure 8. Statistical information on MGOA dataset. It contains 8 *Atomic Tasks*: 'Log ⬢', 'Seed 🌱', 'Dirt ⬢', 'Stone ⬢', 'Iron ⬢', 'Gold 🟨', 'Diamond ◯', 'Redstone 🔴'.

responsible for controlling the movement of agents, such as jumping, forward, back, etc. The mouse movements are responsible for controlling the perspective of agents and the cursor movements when the GUI is opened. The left and right buttons of the mouse are responsible for attacking and
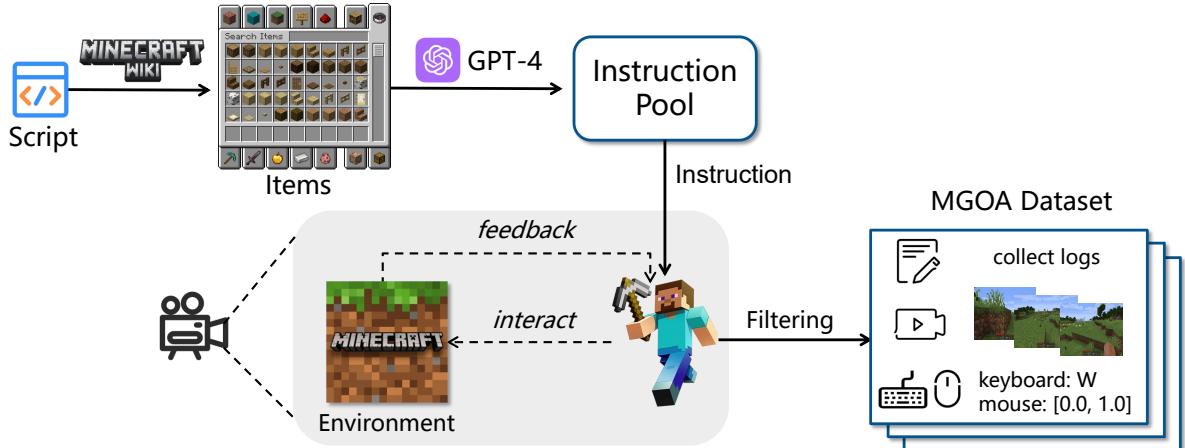
Figure 9. The pipeline for generating the MGOA dataset. First, we extracted item names from the Minecraft Wiki and employed GPT-4 to generate corresponding instructions. These instructions were then provided as input to STEVE-1, enabling it to interact with the environment to accomplish the tasks. During task execution, each observation was paired with its corresponding action, resulting in the creation of goal-observation-action pairs.

using or placing items. In Minecraft, precise mouse movements are important when completing complex tasks that need open inventory or crafting table. In order to achieve both the same action space with MineDojo [10], we abstract the craft and the smelt action into action space. The detailed action space is described in Table 6.

C. MGOA Dataset

In Minecraft, there is still a lack of sufficient high-quality goal-observation-action pairs to support the training of Optimus-2. To address this, we propose an automated dataset construction process aimed at creating high-quality Minecraft Goal-Observation-Action (MGOA) datasets. Through this method. MGOA contains 25,000 videos, providing about 30M goal-observation-action pairs. It contains 8 *Atomic Tasks* across 5 tech levels: 'Log 🪵', 'Seed 🌱', 'Dirt 🟫', 'Stone 🪨', 'Iron ⬜', 'Gold 🟨', 'Diamond 💎', 'Redstone 🔴'. Note that the *Atomic Tasks* in MGOA require minimal steps and can typically be completed within $2 \sim 3$ minutes. For instance, the task 'Iron ⬜' involves mining iron with a stone pickaxe, without the need to gather raw materials to craft the stone pickaxe. The statistics for the MGOA dataset is shown in Figure 8. We provide several examples of the dataset in the MGOA_samples folder within the supplementary materials. We will release this dataset to contribute to the development of open-world agents within the community.

C.1. Dataset Construction

Pipeline. Inspired by Li et al. [24], we employed a prior policy (STEVE-1 [25] in our work) to perform specific tasks

in Minecraft, and recorded the corresponding videos and actions to generate goal-observation-action pairs. As illustrated in Figure 9, we employed a custom script to extract item names from the Minecraft Wiki[3]. Using these item names, we queried GPT-4[4] with a predefined prompt template to generate task instructions, thereby constructing an Instruction Pool. The task instructions from the Instruction Pool serve as input to STEVE-1 [25], enabling it to interact with the environment to complete the tasks. During task execution, each frame and corresponding action were recorded and stored. To expedite data generation, we instantiated multiple policies and used parallelization to quickly produce large amounts of data.

Data Filtering. We judged task success based on environmental feedback. For example, feedback like "obtained new item, diamond axe" indicated that the task "craft a diamond axe" was successfully completed. During the dataset generation process, we observed a significant amount of low-quality video data due to limitations in the policy's ability to follow instructions. Examples of low-quality data included task failures or task completion timeouts. To address this issue, we established two filtering criteria to ensure data quality: (1) only retaining data from successfully completed tasks, and (2) removing data for tasks that lasted longer than 2 minutes. These criteria allowed us to automatically filter out low-quality data, significantly reducing the cost of constructing the dataset. As a result, we obtained a high-quality MGOA dataset consisting of 25,000 samples.

---

[3]https://minecraft.wiki/
[4]https://openai.com/index/gpt-4-research/

C.2. Comparison with Existing Datasets

Previous gameplay videos were primarily obtained through two methods below.

Video Platform: For example, MineDojo [10] collected game videos uploaded by human players on platforms such as YouTube and Twitter, combining the video content with corresponding titles or subtitles to form video-text pairs. However, this dataset lacked recorded actions. To address this, VPT [1] used an Inverse Dynamics Model (IDM) to generate action sequences from the videos. However, the actions predicted by the IDM model are only approximations, which introduces a potential risk of misalignment between the frames and the corresponding actions.

Human Contractors: VPT [1] hired human players to freely explore Minecraft and used the frames and actions to construct a video-action dataset. However, this dataset lacked explicit natural language instructions. To create goal-observation-action pairs, STEVE-1 [25] used GPT-3.5 to generate specific task descriptions based on the gameplay, thereby integrating natural language instructions into the dataset. However, they provide only approximately 32k aligned goal-observation-action pairs, which remains a relatively scarce amount of data.

In addition, some work [32, 43] have utilized GPT-4V to generate image captions, task planning, and reflections, thereby creating image-text pairs that form instruction-following datasets.

Distinct from the aforementioned datasets, the MGOA dataset directly captures agents performing specific tasks, offering clear natural language instructions with a one-to-one correspondence between observations and actions. Furthermore, through rigorous data filtering, redundant action sequences that do not contribute to task completion are excluded from MGOA. In addition, compared to the small-scale goal-observation-action datasets currently available, MGOA offers 25,000 videos, encompassing approximately 30 million goal-observation-action pairs. This dataset is not only significantly larger but also highly scalable in an automated manner.

D. Training Details

D.1. Training Pipeline

One of the key factors in implementing our proposed method lies in the efficient alignment of language with the observation-action sequence, and subsequently translating language space into the action space. To tackle this problem, we adopt a two-phase training approach. First, we align language with the observation-action sequence via behavior pre-training. Then, we transform the language space into the action space through action fine-tuning.

Behavior Pre-training: During the pre-training phase, we integrated the Vision-guided Behavior Encoder into the

Table 7. Hyperparameter setting for pre-training and finetuning.

Hyperparameter	Pre-training	Fine-tuning
Optimizer	AdamW	AdamW
Learning Rate	0.0001	0.00004
Warmup Steps	0	0
Epochs	5	10
Batch Size	32	2048
Num. Frames	5M	80M
LoRA_r	64	64
LoRA_alpha	128	128

model. We used OpenAI Contractor Dataset [1] and a subset of MGOA as training data, which comprised approximately 5,000 videos. To balance efficiency and effectiveness, we freeze the visual encoder, then tune the Vision-guided Behavior Encoder along with a large language model (LoRA [15]). During pre-training, we set the learning rate to 0.0001 and trained for 5 epochs. The hyperparameter settings are shown in Table 7.

Action Fine-tuning: During the fine-tuning phase, we adapted the general MLLM DeepSeek-VL-1.3B [29] to the Minecraft environment, transitioning the model's output space from language to low-level actions. We fine-tuned it using OpenAI Contractor Dataset [1] and MGOA, which comprises approximately 20,000 videos. In this phase, we freeze the Vision-guided Behavior Encoder, visual encoder, and large language model (LoRA), and only fine-tuned the action head. During fine-tuning, we set the learning rate to 0.00004 and train for 10 epochs. The hyperparameter settings are shown in Table 7.

D.2. Implementation Details

For the planner, we follow Li et al. [24], employing Multimodal Hybrid Memory empowered GPT-4V for planning and reflection. For the policy, we train the GOAP through the above pipeline. All experiments were conducted on 8x NVIDIA L40 GPUs. For the MGOA dataset, data collection and filtering were conducted in parallel, taking approximately 7 days. Training required around 2 days, while inference and evaluation on atomic tasks, long-horizon tasks, and open-ended instruction tasks took approximately 4 days.

E. Benchmark

E.1. Evaluation Tasks

The evaluation tasks are divided into three categories: *Atomic Tasks*, *Long-horizon Tasks*, and *Open-ended Instruction Tasks*. For each task, the agent's environment is

(a) chop a tree to get logs

(b) mine dirt

(c) collect seeds

(d) dig down to mine stone
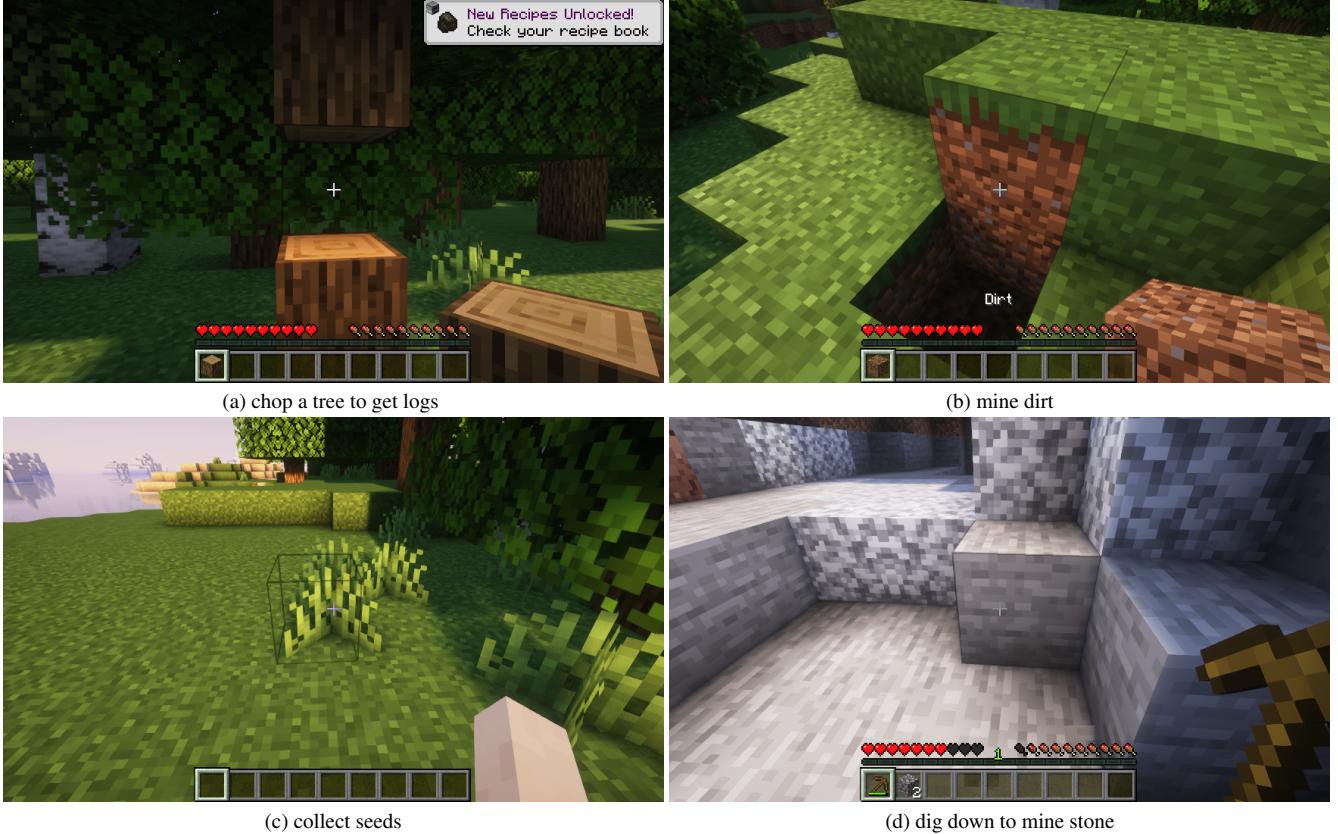
Figure 10. Examples of *Atomic Task*. The agent must follow the instructions to collect resources. These four tasks represent the basic capabilities of the agent. The more resources are collected, the stronger the basic capabilities of the agent will be.

randomly initialized each time, and every task is executed at least 30 times. For *Atomic Tasks*, we follow the setting of prior work [25, 43], which requires the agent to execute the task within 2 minutes. We then report the average reward for the task, defined as the number of items obtained. For *Open-ended Instruction Tasks* and *Long-horizon Tasks*, we report the average success rate (SR) for each task.

Atomic Tasks. As shown in Figure 10, Atomic Tasks are short-term skills in Minecraft, such as "chop a tree to get logs ⬛", "mine dirt ⬛", "collect seeds ⬛", and "dig down to mine stone ⬛", etc.

Long-horizon Tasks. As shown in Figure 11, Long-Horizon Tasks are a sequence of *Atomic Tasks*. For example, "craft an iron sword from scratch" requires completing the atomic tasks of "chop 7 logs", "craft 21 planks", "craft 5 sticks", "craft 1 crafting table", and so on. These *Atomic Tasks* are interdependent, meaning that the failure of any single atomic task will result in the failure of the entire *Long-horizon Task*.

Open-ended Instruction Tasks. Open-Ended Instruction Tasks are not limited to predefined text formats; rather, they involve flexible language directives that prompt the agent to accomplish long-horizon tasks. These tasks evaluate the agent's capacity to interpret and execute instructions expressed in open-ended natural language. We selected Torch ⬛, Rail ⬛, Golden Shovel ⬛, Diamond Pickaxe ⬛, and Compass ⬛ as evaluation tasks. Instruction for each task are shown in Table 8, Table 9, Table 10, Table 11 and Table 12.

E.2. Baselines

In this section, we provide a brief overview of existing Minecraft agents and compare them with our proposed Optimus-2. Current agents can be broadly categorized into two types: policy-based agents and planner-policy agents.

Policy-based Agents. Policy-based agents [1–3, 10, 25] refer to those trained through reinforcement learning or imitation learning, capable of completing atomic tasks within Minecraft. However, due to limitations in instruction understanding and reasoning abilities, they struggle to accomplish long-horizon tasks.

Planner-Policy Agents. Planner-policy agents [20, 24, 32, 41–43] refer to non-end-to-end architectures that utilize a MLLM (Multi-Layered Language Model) as a planner to decompose complex instructions into a sequence of sub-goals executable by a policy. While significant progress has

(a) Chop 7 logs	(b) Craft 21 planks	(c) Craft 5 sticks
(d) Craft 1 crafting table	(e) Craft 1 wooden pickaxe	(f) Mine 11 cobblestone
(g) Craft 1 furnace	(h) Craft 1 stone pickaxe	(i) Dig down more deeper to find iron ore
(j) Mine 2 iron ores	(k) Smelt 2 iron ingots	(l) Craft 1 iron sword
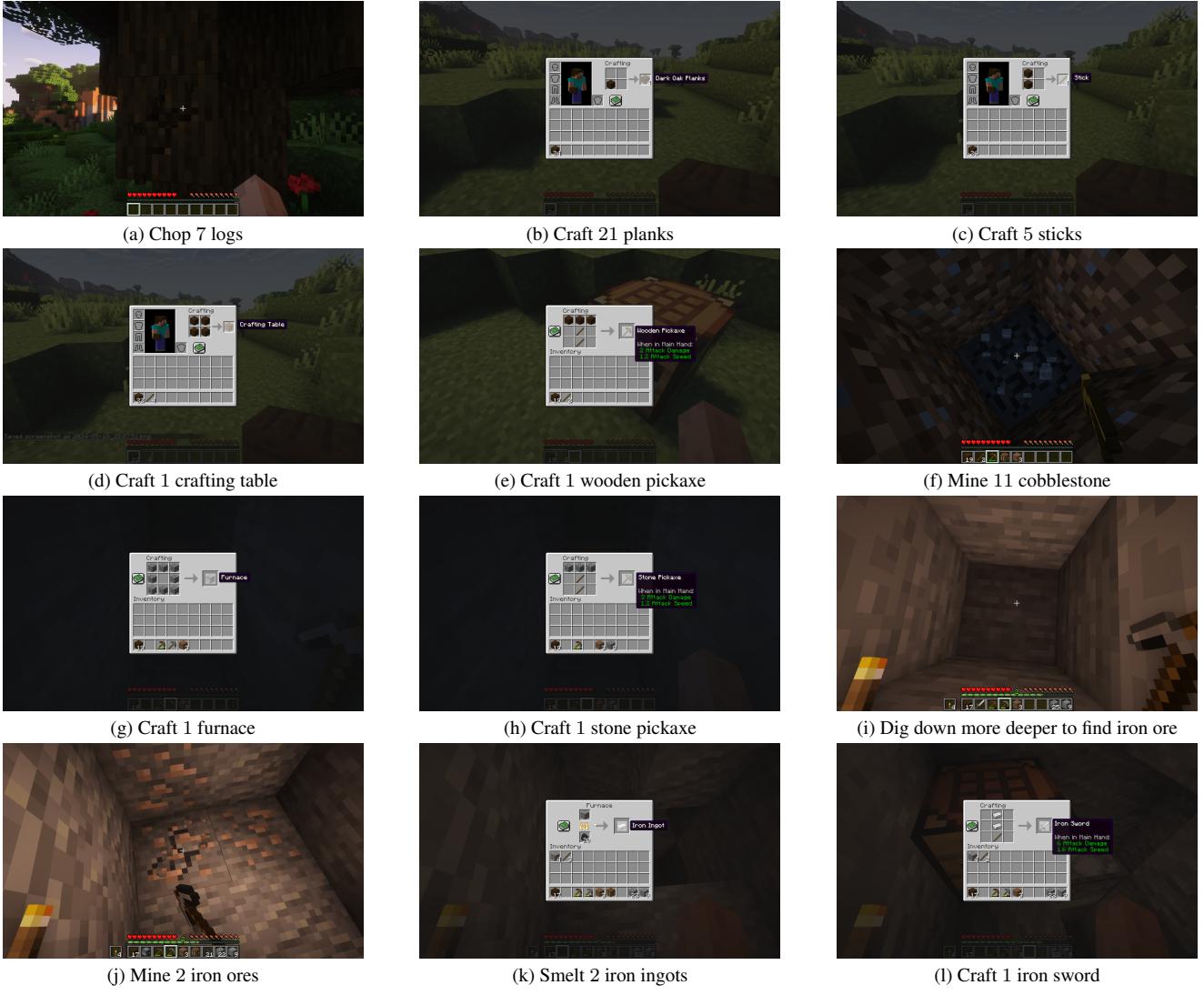
Figure 11. An example of *long-horizon task* "crafting an iron sword". The agent must sequentially complete each atomic task in order to successfully craft the iron sword. Failure in any of the atomic tasks will result in the failure of the entire long-horizon task.

been made, the current performance bottleneck stems from the policy's ability to effectively understand and execute the sub-goals generated by the planner.

Comparison with Existing Agents. As a core contribution of this work, we propose a novel Goal-Observation-Action Conditioned Policy, GOAP. It integrates two key components: an Action-Guided Behavior Encoder for modeling observation-action sequences, and an MLLM for aligning sub-goals with these sequences. Leveraging the MLLM's advanced understanding of open-ended instructions, GOAP demonstrates superior instruction-following capabilities compared to existing policies. On top of GOAP, the proposed agent, Optimus-2, exhibits superior performance in long-horizon tasks, outperforming the current state-of-the-art across all seven task groups.

F. Experimental Results

In this section, we report the experimental results of Optimus-2 on each *Long-horizon task*.

F.1. Results on Long-horizon Task

In this section, we report the results of Optimus-2 on each *Long-horizon Task*, with details including task name, numbers of sub-goals, success rate (SR), and eval times. As shown in Tables 13 and 14, Optimus-2 demonstrates superior performance across all 67 *Long-horizon Tasks*. Since Optimus-2 is randomly initialized in arbitrary environments for each task execution, the experimental results also highlight its generalization capability across diverse environments.

G. Case Study

In this section, we provide additional cases to illustrate the differences in the ability of VPT (text) [1], STEVE-1 [25], and Optimus-2 to perform *Open-ended Instruction Tasks*. We provide different open-ended instructions requiring the agent to perform tasks across various biomes. As shown in Figure 12, Figure 13, and Figure 14, Optimus-2 effectively completes all tasks, while VPT (text) and STEVE-1 fail due to limitations in language understanding and multimodal perception capabilities. Moreover, we provide several demo videos of Optimus-2 performing long-horizon tasks in the `Optimus2_videos` folder within the supplementary materials.

Table 8. Open-ended instruction examples of "Craft a torch"

Planner	Instruction
GPT4V	**Plan 1** 1. First, I need to locate and chop down a couple of trees to collect 4 logs. 2. Next, I should craft these logs into 13 planks. 3. Then, I need to use some of the planks to craft 3 sticks. 4. I will use some of the planks to craft a crafting table. 5. Next, I should craft a wooden pickaxe using some of the planks and sticks. 6. With the wooden pickaxe, I need to find and mine at least 1 coal from a nearby coal deposit or cave. 7. Finally, I can use one stick and one coal to craft a torch. **Plan 2** 1. First, I need to find a tree to gather 4 logs. 2. Next, I have to craft the logs into planks. 3. From the planks, I will craft 3 sticks. 4. I need to craft a crafting table using some of the planks. 5. I should then craft a wooden pickaxe to mine coal. 6. I will locate and mine 1 piece of coal using the wooden pickaxe. 7. Finally, I will use the coal and sticks to craft a torch. **Plan 3** 1. I have to find a tree to get 4 logs. 2. Next, I have to craft planks from the logs. 3. Get sticks from some of the planks. 4. I need a crafting table, which requires planks to craft. 5. I should craft a wooden pickaxe using the crafting table. 6. I need to find and mine some coal using the wooden pickaxe. 7. Finally, with the stick and coal, I can craft a torch.
GLM4V	**Plan 1** 1. Locate and chop down a tree to gather 4 logs. 2. Convert the logs into 13 planks using the crafting menu. 3. Use the planks to craft a crafting table. 4. Craft 3 sticks from the remaining planks. 5. With the crafting table and sticks, craft a wooden pickaxe if one is not already available. 6. Use the wooden pickaxe to dig for coal or kill some Creepers or zombies for coal drops. 7. Once 1 coal is obtained, use the crafting table to combine it with a stick to craft 1 torch. **Plan 2** 1. Locate and chop down a tree to gather the 4 logs required. 2. Convert the logs into 13 planks using the crafting menu, as this will be the material needed for crafting the crafting table and the wooden pickaxe. 3. Use some of the planks to craft a crafting table, which is necessary for advanced crafting recipes. 4. Craft a wooden pickaxe using some of the planks and sticks. The wooden pickaxe will be used to mine coal. 5. Gather 3 sticks, which can be obtained from the logs or by crafting them from planks. 6. Use the wooden pickaxe to mine coal. Only 1 coal is needed to craft a torch. 7. Once all materials are gathered, open the crafting table and combine 1 coal and 1 stick to craft a torch. **Plan 3** 1. Locate and gather 4 logs from the nearest trees. 2. Convert the logs into 13 planks using the crafting menu. 3. Use some planks to craft a crafting table for more complex recipes. 4. Craft a wooden pickaxe using some planks and sticks to gather coal and other resources. 5. Convert some planks into 3 sticks required for crafting torches and the wooden pickaxe. 6. Use the wooden pickaxe to mine coal ore, ensuring to gather at least 1 coal. 7. With all materials gathered, open the crafting table and craft 1 torch using 1 coal and 1 stick.

Table 9. Open-ended instruction examples of "Craft a rail"

Planner	Instruction
GPT4V	**Plan 1** 1. First, locate and punch down a tree to acquire 7 logs. 2. Convert these logs into planks to obtain a total of 25 planks. 3. Use some of these planks to craft 5 sticks. 4. Create a crafting table using 4 planks. 5. Craft a wooden pickaxe using 3 planks and 2 sticks. 6. Use the wooden pickaxe to mine 11 cobblestones. 7. Craft a furnace using 8 cobblestones. 8. Craft a stone pickaxe using 3 cobblestones and 2 sticks. 9. Use the stone pickaxe to mine 6 iron ores. 10. Smelt the iron ores in the furnace to create 6 iron ingots. 11. Finally, craft a rail using the iron ingots and sticks.
	Plan 2 1. I have to find a tree to get 7 logs. 2. Next, I have to craft the logs into 25 planks. 3. Craft 5 sticks from the planks. 4. I need to create a crafting table. 5. Use the crafting table to craft a wooden pickaxe. 6. Use the wooden pickaxe to mine 11 pieces of cobblestone. 7. I'll then craft a furnace using the cobblestone. 8. Craft a stone pickaxe with the cobblestone and sticks. 9. Use the stone pickaxe to mine 6 pieces of iron ore. 10. Smelt the iron ore in the furnace to obtain 6 iron ingots. 11. Finally, use the crafting table to craft a rail using the iron ingots and sticks.
	Plan 3 1. I have to find a tree to get 7 logs. 2. Next, I need to craft planks from the logs. 3. Get sticks from the planks. 4. Craft a crafting table using some of the planks. 5. Craft a wooden pickaxe with sticks and planks to mine stone. 6. Mine 11 cobblestone with the wooden pickaxe. 7. Craft a furnace using the cobblestone. 8. Craft a stone pickaxe using cobblestone and sticks. 9. Use the stone pickaxe to mine 6 iron ore. 10. Smelt the iron ore in the furnace to get 6 iron ingots. 11. Finally, use the iron ingots and sticks to craft a rail.
GLM4V	**Plan 1** 1. Locate and gather 7 logs from nearby trees. 2. Use the crafting menu to convert the logs into 25 planks. 3. Craft 5 sticks from the planks. 4. Place down a crafting table to access more crafting recipes. 5. Use the crafting table to craft a wooden pickaxe. 6. With the wooden pickaxe, mine 11 cobblestones from the ground or from stone blocks. 7. Craft a furnace using the cobblestones. 8. Upgrade the wooden pickaxe to a stone pickaxe by crafting it at the crafting table with the cobblestones. 9. Use the stone pickaxe to mine 6 iron ore from the appropriate rock layers. 10. Smelt the 6 iron ore in the furnace to produce 6 iron ingots. 11. Finally, use the crafting table to craft 1 rail using the 6 iron ingots and 1 stick.
	Plan 2 1. Locate and gather 7 logs from nearby trees. 2. Use the crafting menu to convert the logs into 25 planks. 3. Craft 5 sticks using the planks in the crafting menu. 4. Place a crafting table to enable the crafting of more complex items. 5. Craft a wooden pickaxe using some of the planks and sticks. 6. Use the wooden pickaxe to mine 11 cobblestones from stone outcrops. 7. Construct a furnace using the cobblestones. 8. Upgrade to a stone pickaxe by crafting it with the cobblestones and sticks. 9. Use the stone pickaxe to mine 6 iron ore from iron veins. 10. Smelt the 6 iron ore in the furnace to produce 6 iron ingots. 11. Finally, use the crafting table to craft 1 rail using the iron ingots and sticks.
	Plan 3 1. Find and chop down a tree to gather 7 logs. 2. Use the crafting menu to convert the logs into 25 planks. 3. Craft 5 sticks from the planks. 4. Place down a crafting table to access more crafting options. 5. Craft a wooden pickaxe using some of the planks and sticks. 6. Use the wooden pickaxe to mine 11 cobblestones from stone. 7. Construct a furnace using some of the cobblestones. 8. Upgrade the wooden pickaxe to a stone pickaxe by crafting it with the cobblestones. 9. Search for and mine 6 iron ore blocks with the stone pickaxe. 10. Smelt the iron ore in the furnace to produce 6 iron ingots. 11. Finally, use the crafting table to craft 1 rail using the iron ingots and sticks.

Table 10. Open-ended instruction examples of "Craft a golden shovel"

Planner	Instruction
GPT4V	**Plan 1** 1. Start by finding and chopping down trees to collect 9 logs. 2. Convert the logs into 35 planks using the crafting interface. 3. Use some of the planks to craft 8 sticks. 4. Create a crafting table using some of the planks. 5. Craft a wooden pickaxe. 6. Mine 19 cobblestone. 7. Craft a furnace using 8 of the cobblestones. 8. Craft a stone pickaxe. 9. Mine 3 iron ores using the stone pickaxe. 10. Smelt the iron ore in the furnace. 11. Craft an iron pickaxe using sticks and iron ingots. 12. Locate and mine a gold block using the iron pickaxe. 13. Smelt the gold block in the furnace to get 1 gold ingot. 14. Craft a golden shovel using the gold ingot and sticks. **Plan 2** 1. Start by finding and chopping down trees to collect 9 logs. 2. Use the logs to craft planks. 3. Craft sticks. 4. Use some of the planks to craft a crafting table. 5. Use the crafting table to craft a wooden pickaxe. 6. Use the wooden pickaxe to mine 19 pieces of cobblestone. 7. Use some of the cobblestone to craft a furnace. 8. Use the remaining cobblestone to craft a stone pickaxe. 9. Mine iron ore with the stone pickaxe. 10. Smelt the iron ore in the furnace to obtain 3 iron ingots. 11. Use 2 of the iron ingots and sticks to craft an iron pickaxe. 12. Use the iron pickaxe to mine gold ore or find a gold block; you need at least 1 gold ingot. 13. If mining gold ore, smelt it in the furnace to obtain gold ingots. 14. Use the gold ingot and stick to craft a golden shovel. **Plan 3** 1. Find and cut down trees to collect 9 logs. 2. Use the logs to craft 35 planks. 3. Craft 8 sticks using the planks. 4. Use some of the planks to craft a crafting table. 5. Craft a wooden pickaxe using planks and sticks. 6. Mine cobblestone until you have 19 pieces. 7. Use some cobblestone and the crafting table to craft a furnace. 8. Craft a stone pickaxe using cobblestone and remaining sticks. 9. Use the stone pickaxe to mine 3 pieces of iron ore. 10. Smelt the iron ore in the furnace to get 3 iron ingots. 11. Craft an iron pickaxe. 12. Use the iron pickaxe to mine gold ore to obtain gold blocks. 13. Use 1 gold ingot to craft a golden shovel. 14. use a gold ingot and sticks to craft a golden shovel using the crafting table.
GLM4V	**Plan 1** 1. Find and chop down trees to gather 9 logs. 2. Convert the logs into 35 planks using the crafting menu. 3. Craft 8 sticks from the planks. 4. Build a crafting table to expand the crafting options. 5. Craft a wooden pickaxe with the planks and sticks. 6. Cine cobblestone until you have a total of 19. 7. Craft a furnace using 8 cobblestone. 8. Use the furnace to smelt the cobblestone into 19 stone, then use the crafting table to craft a stone pickaxe with the stone and sticks. 9. Use the stone pickaxe to mine 3 iron ore. 10. Smelt the iron ore in the furnace to produce 3 iron ingots. 11. Craft an iron pickaxe using the iron ingots and sticks at the crafting table. 12. Use the iron pickaxe to mine a gold block. 13. Smelt the gold block in the furnace to get 1 gold ingot. 14. Finally, use the crafting table to craft the golden shovel using 1 gold ingot and 2 sticks. **Plan 2** 1. Find and gather 9 logs from nearby trees. 2. Convert the logs into 35 planks using a crafting table. 3. Craft 8 sticks from the planks. 4. Build a crafting table. 5. Use the crafting table to craft a wooden pickaxe with the planks and sticks. 6. Use the wooden pickaxe to mine 19 cobblestones from stone. 7. Construct a furnace to smelt materials. 8. Craft a stone pickaxe using the cobblestones and sticks. 9. Use the stone pickaxe to mine 3 iron ore from iron deposits. 10. Smelt the 3 iron ore in the furnace to produce 3 iron ingots. 11. Craft an iron pickaxe with the iron ingots and sticks. 12. Use the iron pickaxe to mine a gold block. 13. Smelt the gold block in the furnace to get 1 gold ingot. 14. Finally, craft the golden shovel using the gold ingot and stick in the crafting table. **Plan 3** 1. Find and gather 9 logs from nearby trees. 2. Use the crafting menu to convert the logs into 35 planks. 3. Craft 8 sticks from the planks. 4. Craft a crafting table using some of the planks. 5. Use the crafting table to craft a wooden pickaxe with some of the sticks and planks. 6. Use the wooden pickaxe to mine 19 cobblestones from stone. 7. Craft a furnace using some of the cobblestones. 8. Use the furnace to smelt the cobblestones into 19 stone, then craft a stone pickaxe with the sticks and stone. 9. Mine 3 iron ore from the appropriate rock faces. 10. Smelt the iron ore in the furnace to produce 3 iron ingots. 11. Use the crafting table to craft an iron pickaxe with the iron ingots and sticks. 12. Use the iron pickaxe to mine a gold block. 13. Smelt the gold block in the furnace to produce 1 gold ingot. 14. Finally, use the crafting table to craft the golden shovel with the gold ingot and 2 sticks.

Table 11. Open-ended instruction examples of "Craft a diamond pickaxe"

Planner	Instruction
GPT4V	**Plan 1** 1. First, locate and chop down trees to gather 9 logs. 2. Convert the logs into 35 planks. 3. Use some of the planks to craft 8 sticks. 4. Create a crafting table using 4 planks. 5. With the crafting table, craft a wooden pickaxe using planks and sticks. 6. Use the wooden pickaxe to collect at least 11 cobblestones from stone. 7. With the cobblestones, craft a furnace. 8. Use more cobblestones to craft a stone pickaxe. 9. Use the stone pickaxe to mine 3 iron ores. 10. Smelt the iron ores into 3 iron ingots using the furnace. 11. Craft an iron pickaxe with iron ingots and sticks. 12. Use the iron pickaxe to mine 3 diamonds. 13. Finally, craft the diamond pickaxe using the diamonds and sticks. **Plan 2** 1. Begin by locating a tree to gather 9 logs. 2. Convert the logs into planks. 3. Use some of the planks to craft sticks. 4. Craft a crafting table to facilitate further crafting. 5. Create a wooden pickaxe to start mining stone. 6. Mine 11 cobblestones. 7. Use the cobblestones to craft a furnace. 8. Craft a stone pickaxe to advance to mining iron ore. 9. Locate and mine 3 iron ores using the stone pickaxe. 10. Smelt the iron ores in the furnace to produce 3 iron ingots. 11. Use two of the iron ingots and sticks to craft an iron pickaxe. 12. Locate a diamond ore vein and mine 3 diamonds using the iron pickaxe. 13. Finally, craft the diamond pickaxe with the 3 diamonds and sticks. **Plan 3** 1. I need to find a tree to gather 9 logs. 2. Convert the logs into 35 planks. 3. Use some planks to craft 8 sticks. 4. Craft a crafting table using some of the planks. 5. With the crafting table, make a wooden pickaxe to mine stone. 6. Mine 11 cobblestones using the wooden pickaxe. 7. Craft a furnace using the collected cobblestones. 8. Make a stone pickaxe to efficiently mine iron ore. 9. Find and mine 3 iron ores using the stone pickaxe. 10. Smelt the iron ores in the furnace to produce 3 iron ingots. 11. Craft an iron pickaxe using some planks, sticks, and the iron ingots. 12. Locate a source of diamonds and use the iron pickaxe to mine 3 diamonds. 13. Finally, use the crafting table along with the sticks and diamonds to craft a diamond pickaxe.
GLM4V	**Plan 1** 1. Start by finding trees to gather 9 logs. 2. Convert the logs into 35 planks. 3. Craft 8 sticks from the planks. 4. Build a crafting table to expand crafting options. 5. Use the crafting table to make a wooden pickaxe. 6. With the wooden pickaxe, mine 11 cobblestones. 7. Craft a stone pickaxe using the cobblestones. 8. Construct a furnace to smelt materials. 9. Use the stone pickaxe to mine 3 iron ores. 10. Smelt the iron ores in the furnace to produce 3 iron ingots. 11. Craft an iron pickaxe with the iron ingots. 12. Locate and mine 3 diamonds, which may require exploring caves or mining deep underground with the iron pickaxe. 13. Finally, use the crafting table to combine the diamonds and the sticks to craft a diamond pickaxe. **Plan 2** 1. Locate and chop down trees to gather a total of 9 logs. 2. Use the logs to craft 35 planks, as this will be the base material for several other items. 3. Craft 8 sticks from the planks, which will be needed to create various tools. 4. Craft a crafting table using some of the planks, as this is essential for crafting more complex items. 5. Use the crafting table to craft a wooden pickaxe with the planks and sticks, which will be used to mine cobblestone. 6. Mine 11 cobblestone using the wooden pickaxe. 7. Craft a stone pickaxe with the cobblestone and sticks, as this is more durable and efficient for mining. 8. Construct a furnace using the cobblestone, which will be required to smelt ores into ingots. 9. Mine 3 iron ore with the stone pickaxe. 10. Smelt the iron ore in the furnace to produce 3 iron ingots. 11. Craft an iron pickaxe using the iron ingots and sticks, as this is necessary to mine diamonds. 12. Find and mine diamond ore with the iron pickaxe to gather 3 diamonds. 13. Finally, use the crafting table to craft a diamond pickaxe with the diamonds and sticks, completing the task. **Plan 3** 1. Begin by locating trees to gather 9 logs. 2. Convert the logs into 35 planks. 3. Craft 8 sticks from the planks. 4. Build a crafting table to expand crafting options. 5. Use the crafting table to craft a wooden pickaxe. 6. With the wooden pickaxe, mine 11 cobblestones. 7. Craft a stone pickaxe using the cobblestones. 8. Construct a furnace to smelt materials. 9. Use the stone pickaxe to mine 3 iron ore. 10. Smelt the iron ore in the furnace to produce 3 iron ingots. 11. Craft an iron pickaxe with the iron ingots. 12. Locate and mine diamond ore with the iron pickaxe to obtain 3 diamonds. 13. Finally, use the crafting table to craft a diamond pickaxe with the diamonds.

Table 12. Open-ended instruction examples of "Craft a compass"

Planner	Instruction
GPT4V	**Plan 1** 1. Locate a tree and gather 10 logs. 2. Convert the logs into 38 planks. 3. Craft 8 sticks from the planks. 4. Use some planks to craft a crafting table. 5. With the crafting table, craft a wooden pickaxe. 6. Mine cobblestone until you have 12 pieces. 7. Upgrade your tools by crafting a stone pickaxe. 8. Construct a furnace using some of the cobblestone. 9. Use the stone pickaxe to mine 7 iron ore. 10. Smelt the iron ore in the furnace to produce 7 iron ingots. 11. Craft an iron pickaxe using the iron ingots. 12. Find and mine redstone using the iron pickaxe to obtain 1 redstone dust. 13. Finally, use the remaining iron ingots and redstone dust at the crafting table to craft a compass. **Plan 2** 1. Start by finding and chopping down trees to collect 10 logs. 2. Use the logs to craft 38 planks. 3. Craft sticks using the planks to get 8 sticks. 4. Craft a crafting table from the planks. 5. Use the crafting table to create a wooden pickaxe. 6. Locate and mine 12 pieces of cobblestone using the wooden pickaxe. 7. Craft a stone pickaxe from cobblestone and sticks. 8. Use the cobblestone to craft a furnace. 9. Mine 7 iron ores using the stone pickaxe. 10. Smelt the iron ore in the furnace to get 7 iron ingots. 11. Craft an iron pickaxe using iron ingots and sticks. 12. Search for and mine redstone using the iron pickaxe. 13. Finally, use the iron ingots and redstone to craft a compass. **Plan 3** 1. First, I need to find a tree and collect 10 logs. 2. Convert the logs into planks, ensuring I get at least 38 planks. 3. Use some of the planks to craft sticks, collecting at least 8 sticks. 4. Use some of the planks to create a crafting table. 5. Craft a wooden pickaxe. 6. Locate cobblestone and mine at least 12 pieces using the wooden pickaxe. 7. Use some cobblestone to craft a furnace. 8. Craft a stone pickaxe for improved mining. 9. Search for iron ore and mine at least 7 using the stone pickaxe. 10. Smelt the iron ore in the furnace to produce 7 iron ingots. 11. Craft an iron pickaxe using some of the iron ingots and sticks. 12. Locate a redstone ore and mine it using the iron pickaxe to obtain redstone. 13. Finally, use 4 of the iron ingots and the redstone to craft a compass at the crafting table.
GLM4V	**Plan 1** 1. Locate and gather 10 logs from nearby trees. 2. Convert the logs into 38 planks using the crafting menu. 3. Craft 8 sticks from the planks. 4. Build a crafting table to access more crafting options. 5. Use the crafting table to make a wooden pickaxe. 6. With the wooden pickaxe, mine 12 cobblestones from stone. 7. Craft a stone pickaxe using the cobblestones and the crafting table. 8. Construct a furnace to smelt materials. 9. Use the stone pickaxe to mine 7 iron ores from the appropriate rock formations. 10. Smelt the 7 iron ores in the furnace to produce 7 iron ingots. 11. Craft an iron pickaxe with the iron ingots at the crafting table. 12. Mine for redstone, which can sometimes be found in the ground. 13. Use the crafting table to craft a compass with the iron ingot and redstone. **Plan 2** 1. Locate and chop down trees to gather the required 10 logs. 2. Convert the logs into 38 planks using the crafting menu. 3. Craft 8 sticks from the planks. 4. Use some planks to craft a crafting table. 5. With the crafting table and the sticks and planks crafted, make a wooden pickaxe. 6. Use the wooden pickaxe to mine 12 cobblestone. 7. Craft a stone pickaxe using the cobblestone. 8. Construct a furnace with the cobblestone. 9. Use the stone pickaxe to mine 7 iron ore. 10. Smelt the iron ore in the furnace to produce 7 iron ingots. 11. Craft an iron pickaxe using the iron ingots and the crafting table. 12. Search for redstone; gather at least 1 redstone. 13. Finally, use the crafting table to combine the redstone with the iron ingots to craft a compass. **Plan 3** 1. Find and chop down trees to gather 10 logs. 2. Convert the logs into 38 planks. 3. Craft 8 sticks from the planks. 4. Craft a crafting table using some of the planks. 5. Use the crafting table to craft a wooden pickaxe. 6. With the wooden pickaxe, mine cobblestone until you have at least 12. 7. Upgrade the wooden pickaxe to a stone pickaxe using the cobblestone. 8. Construct a furnace out of the cobblestone. 9. Use the stone pickaxe to mine 7 iron ore. 10. Smelt the 7 iron ore in the furnace to produce 7 iron ingots. 11. Craft an iron pickaxe using the iron ingots and sticks at the crafting table. 12. Search for redstone; you will need at least 1 piece. 13. Finally, use the crafting table to combine the redstone with the iron ingots to craft a compass.

Table 13. The results of Optimus-2 on the Wood Group, Stone Group, and Iron Group. SR denotes success rate.

Group	Task	Sub-Goal Num.	SR	Eval Times
Wood	Craft a wooden shovel	6	100.00	40
	Craft a wooden pickaxe	5	100.00	30
	Craft a wooden axe	5	97.37	38
	Craft a wooden hoe	5	100.00	30
	Craft a stick	4	100	30
	Craft a crafting table	3	93.02	43
	Craft a wooden sword	5	100.00	30
	Craft a chest	4	100.00	30
	Craft a bowl	4	100.00	30
	Craft a ladder	4	100.00	30
Stone	Craft a stone shovel	8	89.47	57
	Craft a stone pickaxe	10	98.00	50
	Craft a stone axe	10	94.44	54
	Craft a stone hoe	8	95.74	47
	Craft a charcoal	9	85.71	42
	Craft a smoker	9	90.00	40
	Craft a stone sword	8	95.45	44
	Craft a furnace	9	94.44	36
	Craft a torch	8	89.36	47
Iron	Craft an iron shovel	13	52.08	48
	Craft an iron pickaxe	13	56.00	50
	Craft an iron axe	13	48.15	54
	Craft an iron hoe	13	56.60	53
	Craft a bucket	13	45.10	51
	Craft a hopper	14	54.90	51
	Craft a rail	13	51.02	49
	Craft an iron sword	12	56.52	46
	Craft a shears	12	48.28	58
	Craft a smithing table	12	53.33	45
	Craft a tripwire hook	13	55.56	45
	Craft a chain	13	52.17	46
	Craft an iron bars	12	51.06	47
	Craft an iron nugget	12	54.55	44
	Craft a blast furnace	14	52.27	44
	Craft a stonecutter	13	52.27	44

Table 14. The results of Optimus-2 on the Gold group, Diamond Group, Redstone Group, and Armor Group. SR denotes success rate.

Group	Task	Sub Goal Num.	SR	Eval Times
Gold	Craft a golden shovel	16	8.93	56
	Craft a golden pickaxe	16	11.29	62
	Craft a golden axe	16	8.93	56
	Craft a golden hoe	16	8.96	67
	Craft a golden sword	16	8.20	61
	Smelt and craft an golden ingot	15	9.68	62
Diamond	Craft a diamond shovel	15	15.91	44
	Craft a diamond pickaxe	15	11.76	34
	Craft a diamond axe	16	11.00	36
	Craft a diamond hoe	15	15.91	44
	Craft a diamond sword	15	11.11	36
	Dig down and mine a diamond	15	11.42	35
	Craft a jukebox	15	13.15	38
Redstone	Craft a piston	16	28.33	60
	Craft a redstone torch	16	27.69	65
	Craft an activator rail	18	25.81	62
	Craft a compass	23	28.36	67
	Craft a dropper	16	30.30	66
	Craft a note block	16	25.40	63
Armor	Craft shield	14	45.16	62
	Craft iron chestplate	14	43.86	57
	Craft iron boots	14	40.35	57
	Craft iron leggings	14	8.57	35
	Craft iron helmet	14	47.46	56
	Craft diamond helmet	17	9.09	33
	Craft diamond chestplate	17	7.89	38
	Craft diamond leggings	17	5.41	37
	Craft diamond boots	17	12.50	40
	Craft golden helmet	17	13.89	36
	Craft golden leggings	17	12.20	41
	Craft golden boots	17	10.26	39
	Craft golden chestplate	17	10.00	40

Figure 12. An illustration of VPT (text) [1], STEVE-1 [25], and Optimus-2 executing the open-ended instruction, "I want to get some logs to craft wooden sword, what should I do first?". Existing policies are limited by their instruction comprehension abilities and thus fail to complete the task, whereas GOAP leverages the language understanding capabilities of the MLLM, enabling it to accomplish the task.

Figure 13. An illustration of VPT (text) [1], STEVE-1 [25], and Optimus-2 executing the open-ended instruction, "I need coal for heating. What should I do?". Existing policies are limited by their instruction comprehension abilities and thus fail to complete the task, whereas GOAP leverages the language understanding capabilities of the MLLM, enabling it to accomplish the task.

Figure 14. An illustration of VPT (text) [1], STEVE-1 [25], and Optimus-2 executing the open-ended instruction, "I want to collect some seeds, Can you help me?". Existing policies are limited by their instruction comprehension abilities and thus fail to complete the task, whereas GOAP leverages the language understanding capabilities of the MLLM, enabling it to accomplish the task.

References

[1] Bowen Baker, Ilge Akkaya, Peter Zhokov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654, 2022. 2, 3, 4, 5, 6, 7, 8, 15

[2] Shaofei Cai, Zihao Wang, Xiaojian Ma, Anji Liu, and Yitao Liang. Open-world multi-task control through goal-aware representation learning and adaptive horizon prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13734–13744, 2023. 2

[3] Shaofei Cai, Bowei Zhang, Zihao Wang, Xiaojian Ma, Anji Liu, and Yitao Liang. Groot: Learning to follow instructions by watching gameplay videos. In *The Twelfth International Conference on Learning Representations*, 2023. 1, 2, 3, 4, 5, 6

[4] Gongwei Chen, Leyang Shen, Rui Shao, Xiang Deng, and Liqiang Nie. Lion: Empowering multimodal large language model with dual-level visual knowledge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26540–26550, 2024. 1

[5] Jingxuan Chen, Derek Yuen, Bin Xie, Yuhao Yang, Gongwei Chen, Zhihao Wu, Li Yixing, Xurui Zhou, Weiwen Liu, Shuai Wang, et al. Spa-bench: A comprehensive benchmark for smartphone agent evaluation. In *ICLR*, 2025. 1

[6] Yihong Chen, Yue Cao, Han Hu, and Liwei Wang. Memory enhanced global-local aggregation for video object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10337–10346, 2020. 3

[7] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. Instructblip: towards general-purpose vision-language models with instruction tuning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 49250–49267, 2023. 3

[8] Ziluo Ding, Hao Luo, Ke Li, Junpeng Yue, Tiejun Huang, and Zongqing Lu. Clip4mc: An rl-friendly vision-language model for minecraft. *arXiv preprint arXiv:2303.10571*, 2023. 2

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 4, 8

[10] 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. 2, 3, 4, 8, 5

[11] Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, et al. Chatglm: A family of large language

models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024. 6

[12] William H Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations. *arXiv preprint arXiv:1907.13440*, 2019. 5, 2

[13] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023. 2

[14] Bo He, Hengduo Li, Young Kyun Jang, Menglin Jia, Xuefei Cao, Ashish Shah, Abhinav Shrivastava, and Ser-Nam Lim. Ma-lmm: Memory-augmented large multimodal model for long-term video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13504–13514, 2024. 3, 5

[15] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 4

[16] Peng Jin, Ryuichi Takanobu, Wancai Zhang, Xiaochun Cao, and Li Yuan. Chat-univi: Unified visual representation empowers large language models with image and video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13700–13710, 2024. 3

[17] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. 8

[18] Sangho Lee, Jinyoung Sung, Youngjae Yu, and Gunhee Kim. A memory network approach for story-based temporal summarization of 360° videos. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1410–1419, 2018. 3

[19] Sangmin Lee, Hak Gu Kim, Dae Hwi Choi, Hyung-Il Kim, and Yong Man Ro. Video prediction recalling long-term motion context via memory alignment learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3054–3063, 2021. 3

[20] Hao Li, Xue Yang, Zhaokai Wang, Xizhou Zhu, Jie Zhou, Yu Qiao, Xiaogang Wang, Hongsheng Li, Lewei Lu, and Jifeng Dai. Auto mc-reward: Automated dense reward design with large language models for minecraft. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16426–16435, 2024. 1, 3, 5

[21] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023. 3

[22] Wei Li, Bing Hu, Rui Shao, Leyang Shen, and Liqiang Nie. Lion-fs: Fast & slow video-language thinker as online video assistant. *arXiv preprint arXiv:2503.03663*, 2025. 3

[23] Xiaojie Li, Jianlong Wu, Shaowei He, Shuo Kang, Yue Yu, Liqiang Nie, and Min Zhang. Fine-grained key-value memory enhanced predictor for video representation learning. In

Proceedings of the ACM International Conference on Multimedia, page 2264–2274. ACM, 2023. 1

[24] 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. *arXiv preprint arXiv:2408.03615*, 2024. 1, 2, 3, 4, 5, 6

[25] Shalev Lifshitz, Keiran Paster, Harris Chan, Jimmy Ba, and Sheila McIlraith. Steve-1: A generative model for text-to-behavior in minecraft. *Advances in Neural Information Processing Systems*, 2023. 1, 2, 3, 4, 5, 6, 7, 15

[26] Shunyu Liu, Yaoru Li, Kongcheng Zhang, Zhenyu Cui, Wenkai Fang, Yuxuan Zheng, Tongya Zheng, and Mingli Song. Odyssey: Empowering agents with open-world skills. *arXiv preprint arXiv:2407.15325*, 2024. 3

[27] Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024. 8

[28] Shaoteng Liu, Haoqi Yuan, Minda Hu, Yanwei Li, Yukang Chen, Shu Liu, Zongqing Lu, and Jiaya Jia. Rl-gpt: Integrating reinforcement learning and code-as-policy. *arXiv preprint arXiv:2402.19299*, 2024. 3

[29] Haoyu Lu, Wen Liu, Bo Zhang, Bingxuan Wang, Kai Dong, Bo Liu, Jingxiang Sun, Tongzheng Ren, Zhuoshu Li, Yaofeng Sun, et al. Deepseek-vl: towards real-world vision-language understanding. *arXiv preprint arXiv:2403.05525*, 2024. 6, 4

[30] Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. Video-chatgpt: Towards detailed video understanding via large vision and language models. *arXiv preprint arXiv:2306.05424*, 2023. 3

[31] Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2661–2670. PMLR, 2017. 2

[32] Yiran Qin, Enshen Zhou, Qichang Liu, Zhenfei Yin, Lu Sheng, Ruimao Zhang, Yu Qiao, and Jing Shao. Mp5: A multi-modal open-ended embodied system in minecraft via active perception. *arXiv preprint arXiv:2312.07472*, 2023. 1, 3, 4, 5

[33] Rui Shao, Xiangyuan Lan, Jiawei Li, and Pong C Yuen. Multi-adversarial discriminative deep domain generalization for face presentation attack detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10023–10031, 2019. 3

[34] Rui Shao, Tianxing Wu, and Ziwei Liu. Detecting and grounding multi-modal media manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6904–6913, 2023. 1

[35] Rui Shao, Tianxing Wu, Jianlong Wu, Liqiang Nie, and Ziwei Liu. Detecting and grounding multi-modal media manipulation and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 3

[36] Rui Shao, Tianxing Wu, Liqiang Nie, and Ziwei Liu. Deepfake-adapter: Dual-level adapter for deepfake detection. *International Journal of Computer Vision*, 2025. 3

[37] Leyang Shen, Gongwei Chen, Rui Shao, Weili Guan, and Liqiang Nie. Mome: Mixture of multimodal experts for generalist multimodal large language models. *Advances in neural information processing systems*, 37:42048–42070, 2025. 1

[38] Enxin Song, Wenhao Chai, Guanhong Wang, Yucheng Zhang, Haoyang Zhou, Feiyang Wu, Haozhe Chi, Xun Guo, Tian Ye, Yanting Zhang, et al. Moviechat: From dense token to sparse memory for long video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18221–18232, 2024. 3

[39] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9 (86):2579–2605, 2008. 8

[40] 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*, 2023. 3

[41] 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*, 2023. 1, 6, 5

[42] Zihao Wang, Shaofei Cai, Anji Liu, Yonggang Jin, Jinbing Hou, Bowei Zhang, Haowei Lin, Zhaofeng He, Zilong Zheng, Yaodong Yang, et al. Jarvis-1: Open-world multitask agents with memory-augmented multimodal language models. *arXiv preprint arXiv:2311.05997*, 2023. 1, 3, 6

[43] Zihao Wang, Shaofei Cai, Zhancun Mu, Haowei Lin, Ceyao Zhang, Xuejie Liu, Qing Li, Anji Liu, Xiaojian Ma, and Yitao Liang. Omnijarvis: Unified vision-language-action tokenization enables open-world instruction following agents. *arXiv preprint arXiv:2407.00114*, 2024. 2, 3, 4, 5, 6

[44] Chao-Yuan Wu, Yanghao Li, Karttikeya Mangalam, Haoqi Fan, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Memvit: Memory-augmented multiscale vision transformer for efficient long-term video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13587–13597, 2022. 3

[45] Qilang Ye, Zitong Yu, Rui Shao, Xinyu Xie, Philip Torr, and Xiaochun Cao. Cat: Enhancing multimodal large language model to answer questions in dynamic audio-visual scenarios. In *European Conference on Computer Vision*, pages 146–164. Springer, 2024. 1

[46] Hang Zhang, Xin Li, and Lidong Bing. Video-llama: An instruction-tuned audio-visual language model for video understanding. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 543–553, 2023. 3

[47] Haoyu Zhang, Meng Liu, Yuhong Li, Ming Yan, Zan Gao, Xiaojun Chang, and Liqiang Nie. Attribute-guided collaborative learning for partial person re-identification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(12):14144–14160, 2023. 1

[48] Haoyu Zhang, Meng Liu, Zixin Liu, Xuemeng Song, Yaowei Wang, and Liqiang Nie. Multi-factor adaptive vision selection for egocentric video question answering. In *Proceedings*

of the 41st International Conference on Machine Learning, pages 59310–59328. PMLR, 2024. 3

[49] Haoji Zhang, Yiqin Wang, Yansong Tang, Yong Liu, Jiashi Feng, Jifeng Dai, and Xiaojie Jin. Flash-vstream: Memory-based real-time understanding for long video streams. *arXiv preprint arXiv:2406.08085*, 2024. 3

[50] Renshan Zhang, Yibo Lyu, Rui Shao, Gongwei Chen, Weili Guan, and Liqiang Nie. Token-level correlation-guided compression for efficient multimodal document understanding. *arXiv preprint arXiv:2407.14439*, 2024. 3

[51] Zhonghan Zhao, Wenhao Chai, Xuan Wang, Li Boyi, Shengyu Hao, Shidong Cao, Tian Ye, Jenq-Neng Hwang, and Gaoang Wang. See and think: Embodied agent in virtual environment. *arXiv preprint arXiv:2311.15209*, 2023. 3

[52] Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Wei-jie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiao-gang Wang, et al. Ghost in the minecraft: Generally capable agents for open-world enviroments via large language models with text-based knowledge and memory. *arXiv preprint arXiv:2305.17144*, 2023. 3