Open-World Multi-Task Control Through
Goal-Aware Representation Learning and Adaptive Horizon Prediction

Shaofei Cai[1,2], **Zihao Wang**[1,2], **Xiaojian Ma**[3], **Anji Liu**[3], **Yitao Liang**[1,4]
Team CraftJarvis

[1]Institute for Artificial Intelligence, Peking University
[2]School of Intelligence Science and Technology, Peking University
[3]Computer Science Department, University of California, Los Angeles
[4]Beijing Institute for General Artificial Intelligence (BIGAI)

{caishaofei,zhwang}@stu.pku.edu.cn,xiaojian.ma@ucla.edu

liuanji@cs.ucla.edu,yitaol@pku.edu.cn

Abstract

We study the problem of learning goal-conditioned policies in Minecraft, a popular, widely accessible yet challenging open-ended environment for developing human-level multi-task agents. We first identify two main challenges of learning such policies: 1) the indistinguishability of tasks from the state distribution, due to the vast scene diversity, and 2) the non-stationary nature of environment dynamics caused by partial observability. To tackle the first challenge, we propose Goal-Sensitive Backbone (GSB) for the policy to encourage the emergence of goal-relevant visual state representations. To tackle the second challenge, the policy is further fueled by an adaptive horizon prediction module that helps alleviate the learning uncertainty brought by the non-stationary dynamics. Experiments on 20 Minecraft tasks show that our method significantly outperforms the best baseline so far; in many of them, we double the performance. Our ablation and exploratory studies then explain how our approach beat the counterparts and also unveil the surprising bonus of zero-shot generalization to new scenes (biomes). We hope our agent could help shed some light on learning goal-conditioned, multi-task agents in challenging, open-ended environments like Minecraft. The code is released at https://github.com/CraftJarvis/MC-Controller.

1. Introduction

Building agents that can accomplish a vast and diverse suite of tasks in an open-ended world is considered a key challenge towards devising generally capable artificial intelligence [2, 3, 6, 35]. In recent years, environments like Minecraft have drawn much attention from the related re-
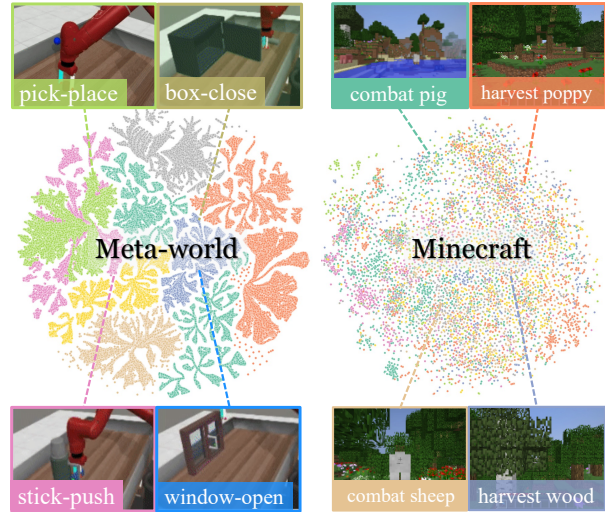

Figure 1. Comparison of states between Meta-world [49] (left) and Minecraft [24] (right) based on t-SNE visualization. The points with the same color represent states from the trajectories that complete the same task. It can be seen that the states are much more distinguishable in terms of tasks in Meta-world than in Minecraft, implying the higher diversity of states and tasks in open worlds like Minecraft over traditional multi-task agent learning environments like Meta-world.

search communities [16, 18–20, 26], since they are not only popular, and widely accessible, but also offer an open-ended universe with myriad of tasks, making them great platforms for developing human-level multi-task agents. Although groundbreaking successes have been observed in many challenging sequential decision-making problems such as Atari [32], Go [39], and MOBA games [13, 44, 45], such successes have not been transferred to those open worlds. To understand the gap and design corresponding solutions, we need to first understand the distinct challenges

brought by these environments. Let's take Minecraft [24] as an example: there are over twenty types of landscapes ranging from flat lands like Savannah and desert to rough mountains with forests and caves. These diverse landscapes also enable countless tasks that could be achieved by the agents: mining, harvesting, farming, combating, constructing, etc. Compared to canonical agent learning environments like Go [39], Atari [32], and robotic control suite [41, 43, 48], Minecraft provides a substantially more diverse distribution of states thanks to the rich scenes and tasks built with the game, making it exceptionally difficult to extract the pivotal task-relevant visual state representations for goal-conditioned policies. To help our readers understand the significance of this challenge, we visualize the states from trajectories that complete some tasks in Minecraft and Meta-world [48] (a popular multi-task learning environment but with fewer states and tasks) in Fig. 1. States of different tasks are annotated with different colors. Clearly, the states in Minecraft are much less distinguishable in terms of tasks than in Meta-world. Therefore goal-conditioned policies are more likely to struggle in mapping those states and tasks (served as goals) to actions.

Another grand challenge in an open-ended environment like Minecraft hails from the setting of such games, where an agent can only have very limited observations of the world. For example, in MineDoJo [16] (a recent agent benchmark built on Minecraft), the observation space comprises a first-person view image and a list of possessed items. However, many more aspects of the surroundings remain hidden from the agents. That is, the agent now has to work with a **partially observable environment**. A plague embedded with such an environment is *non-stationary dynamics*, which makes it almost impossible to predict what will happen next. Therefore, the distances from states to the current goal become much less clear due to the world uncertainty, leading to less distinguishable states in terms of goal completeness and more faulty decisions emitted by the goal-conditioned policies.

This paper aims at mitigating both aforementioned challenges that emerge from most open-world environments. First, we observe that the architecture of the policy network is crucial to learning goal-relevant visual state representations that allow goal-conditioned actions in domains with low inter-goal state diversity (cf. Fig. 1). To this end, we propose Goal-Sensitive Backbone (GSB), which enables effective learning goal-conditioned policies over 20 tasks in the Minecraft domain. Next, to mitigate the challenge posed by the partially observed and non-stationary environment, we introduce horizon as an extra condition for the policy and a corresponding horizon prediction module. Specifically, the policy is also *explicitly* conditioned on the remaining time steps till achieving certain goals (i.e., distance-to-goal). We find it significantly boosts the performance of

our agents in open-world multi-task domains. However, the ground-truth distance-to-goal is unavailable during evaluation. To fix this problem, we train a horizon prediction module and feed the estimated distance-to-goal to the horizon commanding policy in evaluation. This leads to a 27% gain in average success rate under the multi-task settings.

We evaluate the proposed approaches based on the simple yet effective behavior cloning algorithm [10]. The experiments are conducted in three common biomes. In multi-task settings, our proposed method outperforms the baseline in terms of success rate and precision by a large margin. It also achieves consistent improvement in single-task settings. Our ablation and exploratory studies then explain how our approach beat the counterparts and also unveil the surprising bonus of zero-shot generalization to new scenes (biomes).

To summarize, targeting two identified challenges distinct to open worlds, our contributions are threefold:

- We propose Goal-Sensitive Backbone (GSB), a neural network that enables effective learning goal-relevant visual state representations at multiple levels for goal-conditioned policies, aiming at addressing the challenge of diverse state distribution in open-ended environments.

- We further introduce adaptive horizon prediction to explicitly condition the policy on the distance from the current state to the goal, yielding much better performances in a partially observable open-ended environment with non-stationary dynamics.

- We conduct extensive studies on the popular yet challenging Minecraft domain with baselines and our proposed method. The results demonstrate superior advantages of our approach over the counterparts in terms of both success rate and precision of task completion.

2. Preliminaries

Goal-conditioned policy, as its name suggests, is a type of agent's policy π for decision-making that is conditioned on goals besides states. Specifically, we denote $\pi(a|s, g)$ as a goal-conditioned policy that maps the current state s and goal g to an action a. Compared to the canonical formulation of policy where the goal is absent, the goal-conditioned policy offers flexibility of learning *multi-task* agent as it allows different behaviors for different tasks by simply altering the goal. There are multiple ways to specify the goal, e.g., natural language instructions [2] and goal images [36]. **Goal-conditioned imitation learning** is a simple yet effective way to learn goal-conditioned policies. Specifically, $\pi(a|s, g)$ is optimized by imitating the demonstrations $\mathcal{D}$, where $\mathcal{D} = \{\tau^1, \tau^2, \tau^3, \dots\}$ is a collection of trajectories τ^i. A trajectory is a sequence of states, actions, and goals, defined as $\tau^i = \{(s_t^i, a_t^i, g^i)\}_{t=0}^T$, where T is the trajectory length. The imitation learning objective is to maximize the
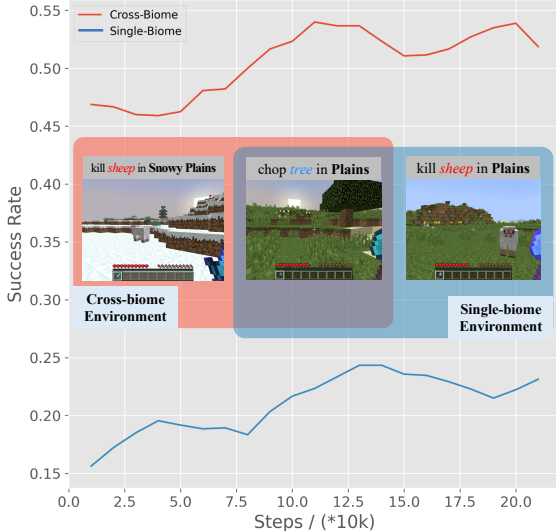
Figure 2. Demonstrations of the cross-biome environment and the more challenging single-biome environment. The challenge comes from the fact that the agent needs to learn diverse behaviors in similar states conditioned on different goals.

likelihood of the action in demonstrations when attempting to reach the desired goal

$$J_{IL}(\pi) = \mathbb{E}_{\tau \sim \mathcal{D}} \big[\sum\nolimits_{t=0}^{T} \log \pi(a_t|s_t, g) \big]. \qquad (1)$$

Notation. At each timestep, our architecture takes in a tuple $(s_t, a_t, h_t, g, a_{t-1})$ as the input, where $s_t = \{o_t^I, o_t^E\}$, o_t^I is the raw image observation, o_t^E is the extra observation provides by the environments. h_t comes from the demonstration. $\tilde{h}_t$ and $\tilde{a}_t$ are the predicted horizon and action, respectively. For simplicity, we also use the same symbols (o_t^E, g, a_{t-1}) to represent their embeddings.

3. Method

In this section, we describe the proposed algorithm for learning goal-conditioned policies that are capable of completing various preliminary tasks in open-world domains. First, we revisit and provide a detailed illustration of the identified challenges in open-world domains (§3.1). Aiming at solving these challenges, we proceed to introduce the proposed goal-sensitive backbone (§3.2) and adaptive horizon prediction module (§3.3). Finally, we provide an overview of the proposed method in Section 3.4.

3.1. Challenges

As demonstrated in Section 1, the **first** major challenge of open-world environments is the indistinguishability of states in terms of different goals (cf. Fig. 1). That is, it is often hard to identify the task/goal by looking at individual states. Compared to environments with clear goal indicators

in their states, agents in open-world domains need to learn goal-conditioned diverse behaviors under similar states.

This challenge can be reflected by the illustrative experiment in Fig. 2. Two multi-task environments are created based on the Minecraft domain. Both environments consist of two preliminary tasks: collect logs and hunt sheep, where the former can be done by chopping trees and the latter requires the agent to slaughter sheep. Both tasks require the agent to first locate and approach the corresponding target. As shown in Fig. 2 (center), in the single-biome environment (blue blob in Fig. 2), the agent is tasked to collect logs and hunt sheep both inside a randomly generated plain area with grass, trees, and various mobs. In contrast, in the cross-biome environment (red blob in Fig. 2), whenever the agent is tasked to hunt sheep, it is spawned randomly in a snowy plain. Although different in visual appearance, snowy plains and plains have very similar terrains, so the difficulty of each task in the cross-biome environment is similar to its counterpart in the single-biome environment. The main consequence of this change is that the agent can determine its goal by solely looking at the current state, which mimics the setting of Meta-World in Fig. 1(left).

We collect demonstrations by filtering successful trajectories played by VPT [4] (see §4.1 for more details) and use behavior cloning to train multi-task policies on both environments. Perhaps surprisingly, as shown in Fig. 2, despite the minor difference, performance in the single-biome environment is significantly weaker than in the cross-biome one. This clearly demonstrates that the common practice of directly concatenating observation features and goal features suffer from learning diverse actions (e.g., locate trees, find sheep) given similar observations. In contrast, in the cross-biome environment, the difficulty of the two tasks fundamentally remains the same, yet the agent only needs to learn a consistent behavior in each biome (i.e., plains and snow fields). This alleviates the need to learn goal-conditioned diverse behaviors in similar states and leads to a better success rate.

The **second** key challenge comes from the partial observability of the game and non-stationary environment dynamics. Specifically, in Minecraft, the biome and mobs surrounding the agent are generated procedurally and randomly after each reset. Further, only a small fraction of the whole terrain is visible to the agent in one observation, leading to more uncertainty of the world. From the perspective of learning goal-conditioned policies, the distances from states to the current goal will become much less clear compared to canonical learning environments like Atari [12]. We refer to Appendix B for more discussion on this. Since the goal-conditioned policies also rely on distinguishable states in terms of goal completeness, they're more likely to make wrong decisions as a result of world uncertainty.
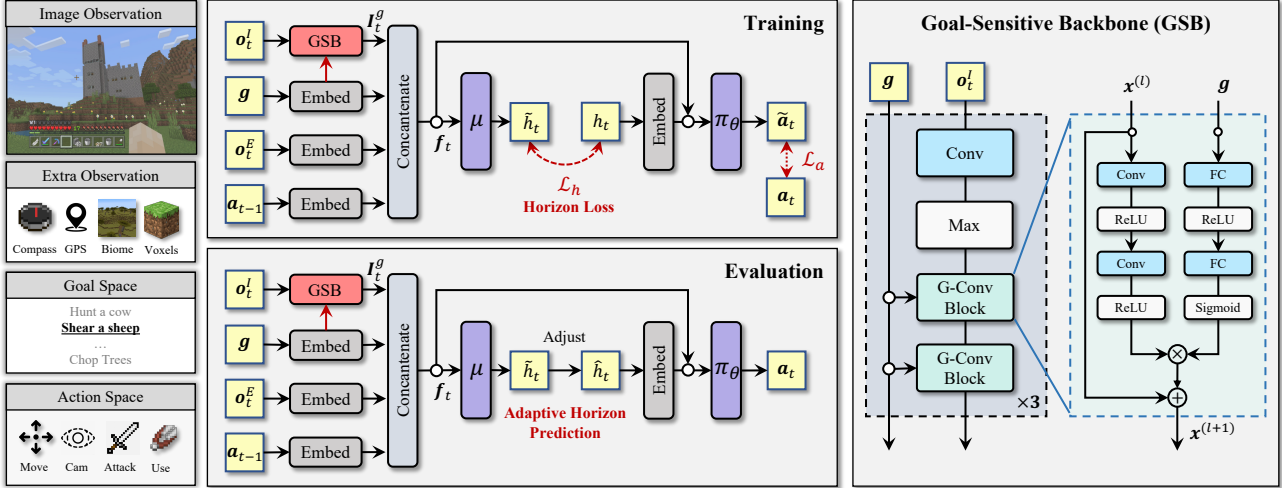
3

Figure 3. **Our Goal-conditioned Policy Architecture**. Our contributions are in red and purple. **Right:** The *goal-sensitive backbone* (GSB) is a key component to incentivize goal-condition behaviors. It consists of a stack of g-conv blocks. It takes the image observation $\boldsymbol{o}_t^I$ and the goal embedding $\boldsymbol{g}$ as input, and outputs the goal-attended visual representation $\boldsymbol{I}_t^g$. The multimodal joint representation $\boldsymbol{f}_t$ is the concatenation of visual representation $\boldsymbol{I}_t^g$, goal embedding $\boldsymbol{g}$, extra observation embedding $\boldsymbol{o}_t^E$ and previous action embedding $\boldsymbol{a}_{t-1}$. The horizon prediction module μ uses it to predict the horizon $\tilde{h}_t$ while the horizon commanding policy π_θ uses it to predict the action $\tilde{\boldsymbol{a}}_t$. **Top:** During the training, the predicted horizon $\tilde{h}_t$ is only used to compute the horizon loss $\mathcal{L}_h$. The policy is conditioned on h_t that comes from the demonstration. **Bottom:** During the evaluation, the policy is conditioned on the predicted horizon $\tilde{h}_t$ which needs to be adjusted.

3.2. Incentivize Goal-Conditioned Behavior with Stacked Goal-Sensitive Backbone

As elaborated in Section 3.1, learning goal-conditioned policies becomes extremely hard when states collected from trajectories that accomplish different tasks are indistinguishable. While certain algorithmic design choices could improve multi-task performance in such open-world environments, we find that the structure of the policy network is a key factor towards higher episode reward. Specifically, we observe that existing CNN-based backbones can excel at completing many single tasks (e.g., hunt cow, collect stone), but struggle to learn goal-conditioned behavior when training on the tasks in a goal-conditioned manner. This motivates the need to properly fuse goal information into the network. Despite the existence of various feature fusion approaches such as concatenation and Bilinear layers [27], they all perform poorly even with a moderate number of tasks. This motivates the need to carry goal information into multiple layers of the network. Specifically, we propose goal-sensitive backbone (GSB), which effectively blends goal information to the state features at multiple levels. As shown in Fig. 3 (right), GSB is composed with multiple goal convolution blocks (g-conv block), which are obtained by augmenting the vanilla convolution block with a goal branch. Functionally, it can provide deep feature fusion between multi-level visual features and the goal information. As we will proceed to show in Section 4.3, adding GSB can lead to significant performance boost in multi-task environments. The g-conv block processes its input visual features $\boldsymbol{x}^{(l)} \in \mathbb{R}^{C \times H \times W}$ with two convolution layers

$$\hat{\boldsymbol{x}}^{(l)} = \mathrm{ReLU}(\mathrm{Conv}(\mathrm{ReLU}(\mathrm{Conv}(\boldsymbol{x}^{(l)})))). \quad (2)$$

Meanwhile, it maps the goal embedding $\boldsymbol{g}$ to the same feature space as the intermediate features $\hat{\boldsymbol{x}}^{(l)}$ with two fully-connected layers, decribed as

$$\hat{\boldsymbol{g}}^{(l)} = \mathrm{FC}(\mathrm{ReLU}(\mathrm{FC}(\boldsymbol{g}))). \quad (3)$$

The goal feature $\hat{\boldsymbol{g}}^{(l)}$ is then used to modulate the intermediate features $\hat{\boldsymbol{x}}^{(l)}$ channel-wise. By adding a residual connection [21], the output feature $\boldsymbol{x}^{(l+1)}$ is expressed by

$$\boldsymbol{x}^{(l+1)} = \sigma(\hat{\boldsymbol{g}}^{(l)}) \odot \hat{\boldsymbol{x}}^{(l)} + \boldsymbol{x}^{(l)}, \quad (4)$$

where $\sigma(\cdot)$ is the sigmoid function and $\odot$ is the element-wise product. This channel-wise modulation encourages the module to focus on goal-specific regions and discard the background information by adaptively weighing the channel importance. We highlight that the g-conv block can be plugged into any convolution backbone to improve its capability of extracting goal-aware visual features. The proposed goal-sensitive backbone is constructed by replacing 6 convolution blocks of the widely-adopted Impala CNN [14] to g-conv blocks. In our experiments, a GSB is used to compute goal-conditioned state features $\boldsymbol{I}_t^g = \mathrm{GSB}(\boldsymbol{o}_t^I, \boldsymbol{g})$. Such an idea of fusing condition information into the backbone layer by layer was also used by some prior works [5, 22, 33, 34]. Here, we demonstrate that it works in a critical role for open-world multi-task control.

3.3. Combat World Uncertainty with Adaptive Horizon Prediction

To address the challenge brought by the uncertainty of the world, we need to ensure the goal-conditioned policies to be more aware of goal-completeness given the current state. We observe that conditioning the policy additionally on the number of remaining steps toward achieving a goal, i.e., distance-to-goal, or **horizon**, can significantly improve the accuracy of predicted actions on held-out offline datasets [17, 37]. Here, we define the horizon $h_t := T - t$, where T is the trajectory length, as the remaining time steps to complete the given goal. This motivates the design of a horizon commanding policy $\pi_\theta : \mathcal{S} \times \mathcal{G} \times \mathcal{H} \to \mathcal{A}$ that takes a state s, a goal g, and a horizon h as inputs and outputs an action a. A key problem of the horizon commanding policy is that it cannot be directly used for evaluation: during gameplay, horizon is unknown as it requires completing the whole trajectory. To fix this problem, we introduce an additional horizon prediction module, which estimates the horizon given a state s and a goal g. Combining the two modules together, we can apply the fruitful horizon commanding policy during gameplay.

Both modules can be trained efficiently with dense supervision. Specifically, the horizon commanding policy π_θ can be learned by any policy loss specified by RL algorithms. For example, when behavior cloning is used, π_θ can be optimized by minimizing the loss

$$\mathcal{L}_a = -\log \pi_\theta(\boldsymbol{a}_t | h_t, \boldsymbol{f}_t), \tag{5}$$

where $\boldsymbol{f}_t$ is the joint representation of the state and goal embedded by a neural network (see §3.4). The horizon prediction module is trained by a supervised learning loss

$$\mathcal{L}_h = -\log \mu(h_t | \boldsymbol{f}_t), \tag{6}$$

where μ is a network that predicts the horizon.

During the evaluation, after computing the embedding $\boldsymbol{f}_t$ for s_t and g, the horizon prediction module μ is first invoked to compute an estimated horizon $\tilde{h}_t = \mu(\boldsymbol{f}_t)$. This predicted horizon can then be fed to the horizon commanding policy to compute the action distribution $\pi_\theta(\boldsymbol{a}_t | \tilde{h}_t, \boldsymbol{f}_t)$. In practice, we observe that feeding an adaptive version of $\tilde{h}_t$, defined as $\hat{h}_t := \max(\tilde{h}_t - c, 0)$ (c is a hyperparameter), to π_θ leads to better performance. We hypothesize that this advantageous behavior comes from the fact that by supplying the adaptive horizon $\hat{h}_t$, the agent is encouraged to choose actions that lead to speedy completion of the goal. The effectiveness of the adaptive horizon will be demonstrated in Section 4.3.

3.4. Model Summary

As shown in Fig. 3, our model sequentially connects the proposed goal-sensitive backbone, horizon prediction module, and horizon commanding policy. At each time step
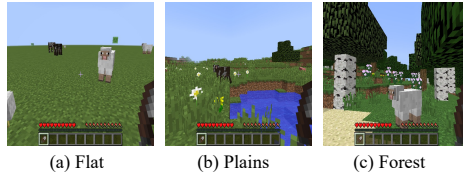

(a) Flat   (b) Plains   (c) Forest
Figure 4. Snapshots of the RGB camera view in three biomes.

t, the image observation and goal information are first fed forward into the goal-sensitive backbone to compute goal-aware visual feature $\boldsymbol{I}_t^g$. The visual feature is then fused with additional input information including the extra observation embedding $\boldsymbol{o}_t^E$, the goal embedding $\boldsymbol{g}$, and the previous action embedding $\boldsymbol{a}_{t-1}$ by concatenation and a feed-forward network:

$$\boldsymbol{f}_t = \text{FFN}([\boldsymbol{I}_t^g \parallel \boldsymbol{o}_t^E \parallel \boldsymbol{g} \parallel \boldsymbol{a}_{t-1}]). \tag{7}$$

Then, $\boldsymbol{f}_t$ is input to the horizon prediction module to predict horizon $\tilde{h}_t = \mu(\boldsymbol{f}_t)$. And the horizon commanding policy takes in the horizon and features $\boldsymbol{f}_t$ to compute the action. When trained with behavior cloning, the overall objective function is $\mathcal{L} = \mathcal{L}_a + \mathcal{L}_h$. During the evaluation, the adaptive horizon $\hat{h}_t$ is fed to the horizon commanding policy in replacement of $\tilde{h}_t$.

4. Experiments

This section analyzes and evaluates the proposed goal-sensitive backbone and the adaptive horizon prediction module in the open-world domain Minecraft. To minimize performance variation caused by the design choices in RL algorithms, we build the proposed method on top of the simple yet effective behavior cloning algorithm. In Section 4.1, we first introduce three suites of tasks; the agent is asked to collect and combat various target objects/mobs with indistinguishable states conditioned on different goals (challenge #1) and non-stationary environment dynamics (challenge #2). Single-task and multi-task performance on the benchmarks is evaluated and analyzed in Section 4.2, and ablation studies are conducted in Section 4.3. Finally, we unveil the surprising bonus of zero-shot generalization to new scenes and tasks in Section 4.4.

4.1. Experimental Setup

Environment and task. To best expose the challenges described in Sections 1 and 3.1, a key design principle of our benchmark environments is to task the agent to complete multiple preliminary tasks in similar yet highly randomized scenes. By specifying the biome that surrounds the agent, Minecraft provides a perfect way to create such environments. Specifically, as shown in Fig. 4, every biome has unique and consistent observations; randomness comes from the fact that the terrain is generated randomly in each episode. To evaluate the scalability of the proposed method in terms of the number of tasks, we choose **Plains** and

Forest, the two most common biomes that contain a large number of resources and mobs.

In addition to the two challenges, **Plains** and **Forest** also add unique difficulties to learning goal-conditioned policies. Specifically, although we have better views in **Plains**, the resources/targets are located further away from the agent and require more exploration. In contrast, there exist more occlusions and obstacles in **Forest**.

The **Plains** benchmark consists of four tasks: harvest `oak wood` (🪵), and Combat `sheep` (🐑), `cow` (🐄), `pig` (🐷). In the **Forest** benchmark, the agent is tasked to complete thirteen tasks: combat `sheep` (🐑), `cow` (🐄), `pig` (🐷), harvest `dirt` (🟫), `sand` (🟨), `oak wood` (🪵), `birch wood` (🪵), `oak leaves` (🌳), `birch leaves` (🌳), `wool` (⬜⬛), `grass` (🌱), `poppy` (🌷), `orange tulip` (🌷).

In addition to the above two benchmarks, we also test the agent on a "hunt animals" benchmark based on the **Flat** biome, which contains a flattened world. Specifically, the agent needs to combat `sheep` (🐑), `cow` (🐄), `pig` (🐷), `spider` (🕷), `polar bear` (🐻), `chicken` (🐔), `donkey` (🐴), `horse` (🐴), `wolf` (🐺), `llama` (🦙), `mushroom cow` (🐄) in the **Flat** environment. Compared to other benchmarks, the challenge of **Flat** comes from the fact that the mobs are constantly wondering around, which makes it hard to locate and approach the correct target.

We adopt the original observation space provided by MineDoJo [16], which includes a RGB camera-view, yaw/pitch angle, GPS location, and the type of 3×3 blocks surrounding the agent. We discretize the original multi-discrete action space provided by MineDojo into 42 discrete actions. Details are included in Appendix A.1.

Data collection pipeline. One significant downside of behavior cloning algorithms is the need for high-quality and densely-labeled trajectories, which often requires enormous human effort to collect. To mitigate this problem, we collect goal-conditioned demonstrations by filtering successful trajectories from gameplays by pretrained non-goal-conditioned policies. Specifically, we adopt Video Pre-Training (VPT) [4], which is trained on tremendous amount of non-goal-conditioned gameplays. We rollout the VPT policy in the three benchmarks and record all episodes that accomplishes any of the defined goals. These trajectories are then converted to a goal-conditioned demonstration dataset. Please refer to Appendix A.2 for detailed settings and efficiency analysis of our data collection pipeline.

Evaluation. During the evaluation, the maximum episode length is set to 600, 600, and 300 on the **Flat**, **Plains** and **Forest** benchmarks, respectively. **Plains** and **Forest** are given more time steps since, in these environments, the agent needs more time to locate and approach the target. We use *Success Rate* and *Precision* as our evaluation metrics. A gameplay is successful if the agent completes the goal within the episode. Precision is defined as the number of times the specified goal is achieved divided by the total number of goals completed in an episode. It measures how well the agent can be aware of the specified goal, instead of simply accomplishing any goal during gameplay.

4.2. Experimental Results

We first focus on the simpler single-task learning setting in order to isolate the challenge introduced by non-stationary dynamics and partial observability (§4.2.1). We then examine whether the proposed method can better address both challenges by examining its multi-task performance (§4.2.2).

4.2.1 Single task experiments

We select three typical tasks, i.e., harvest log, hunt cow, and hunt sheep, from the **Plains** benchmark for single-task training. We compare the proposed method against the following baselines. First, MineAgent [16] is an online RL algorithm that leverages pretrained state representations and dense reward functions to boost training. BC (VPT) [4], BC (CLIP) [16], and BC (I-CNN) [14] are variants of the behavior cloning algorithm that use different backbone models (indicated in the corresponding brackets) for state feature extraction. The backbones are finetuned with the BC loss (see Appendix A.3 for more details).

Results are reported in Table 1. First, we observe that even the individual tasks are extremely challenging for online RL algorithms such as MineAgent, even its networks are pretrained on Minecraft data. We attribute this failure to its inconsistent dense reward when facing a hard-exploration task (e.g., the additional provided reward is not consistently higher when the agent is moving closer to a target object). Next, compared to BC (I-CNN) that uses a randomly initialized impala CNN model, the Minecraft-pretrained backbones in BC (VPT) and BC (CLIP) do not bring any benefit. This could be caused by the lack of plasticity, i.e., the ability to learn in these well-trained models, echoing similar findings in computer vision and RL [11]. Finally, our approach outperforms all baseline methods, especially in terms of precision. This demonstrates that our method is more robust against non-stationary dynamics and partially observable observations.

4.2.2 Multi-task experiments

We move on to evaluate the proposed method on the three multi-task benchmarks introduced in Section 4.1. The baseline includes three behavior cloning methods (we use "MT-BC" as an abbreviation of multi-task behavior cloning). We also include two variations of our method: one without the goal-sensitive backbone, and the other without the adaptive

Table 1. Results of **single-goal** tasks (§4.2.1) on **Plains**.

Method	Success Rate (%)			Precision (%)		
MineAgent [16]	$00_{\pm00}$	$01_{\pm00}$	$01_{\pm00}$	–	–	–
BC (CLIP) [16]	$18_{\pm06}$	$26_{\pm05}$	$25_{\pm06}$	$51_{\pm08}$	$43_{\pm08}$	$44_{\pm05}$
BC (VPT) [4]	$22_{\pm08}$	$27_{\pm06}$	$22_{\pm06}$	$58_{\pm09}$	$46_{\pm05}$	$42_{\pm05}$
BC (I-CNN) [14]	$45_{\pm05}$	$46_{\pm04}$	$48_{\pm07}$	$\mathbf{86_{\pm05}}$	$55_{\pm12}$	$45_{\pm07}$
Ours	$\mathbf{50_{\pm07}}$	$\mathbf{58_{\pm10}}$	$\mathbf{60_{\pm08}}$	$83_{\pm10}$	$\mathbf{75_{\pm10}}$	$\mathbf{75_{\pm06}}$

Table 2. Results of **multi-goal** tasks (§4.2.2) on three biomes.

Method	Avg. Success Rate (%)			Avg. Precision (%)		
	Plains	Flat	Forest	Plains	Flat	Forest
MT-BC (VPT) [4]	$25_{\pm06}$	$17_{\pm05}$	$15_{\pm04}$	$22_{\pm05}$	$17_{\pm03}$	$14_{\pm04}$
MT-BC (CLIP) [16]	$22_{\pm05}$	$14_{\pm03}$	$14_{\pm03}$	$23_{\pm04}$	$15_{\pm03}$	$13_{\pm03}$
MT-BC (I-CNN) [14]	$25_{\pm02}$	$18_{\pm02}$	$15_{\pm03}$	$23_{\pm04}$	$14_{\pm02}$	$13_{\pm03}$
MT-BC (w/ GSB)	$32_{\pm05}$	$36_{\pm03}$	$19_{\pm05}$	$43_{\pm06}$	$36_{\pm02}$	$17_{\pm03}$
Ours (I-CNN)	$31_{\pm06}$	$31_{\pm04}$	$18_{\pm02}$	$22_{\pm03}$	$28_{\pm04}$	$15_{\pm04}$
Ours (w/ GSB)	$\mathbf{55_{\pm09}}$	$\mathbf{57_{\pm09}}$	$\mathbf{30_{\pm06}}$	$\mathbf{70_{\pm09}}$	$\mathbf{50_{\pm06}}$	$\mathbf{29_{\pm06}}$

horizon prediction module. Results on the **Plains**, **Flat**, and **Forest** environments are reported in Table 2, respectively. First, we observe that our method significantly outperforms all baselines in terms of both success rate and precision in all three benchmarks. Moreover, scaling up the number of tasks does not necessarily deteriorate the performance of our method. Specifically, we compare the average success rate on the **Plains** and **Flat** benchmark, which contain 4 and 9 tasks, respectively. While the baselines struggle to maintain their success rate on the **Flat** environment, our approach is capable of maintaining high performance despite the increased number of tasks. Putting together, results on multi-task benchmarks clearly demonstrate the superiority of our method when facing open-world environments with the two elaborated challenges (cf. §3.1).

4.3. Ablation Study

Ablation study on goal-sensitive backbone. To examine the effectiveness of our proposed goal-sensitive backbone, we compare the following two groups of architectures: 1) **Ours** (I-CNN) v.s. **Ours** (w/ GSB), 2) MT-BC (I-CNN) v.s. MT-BC (w/ GSB). The key distinction between the groups is whether the backbone employs a standard Impala CNN or a goal-sensitive backbone. As depicted in Table 2, our findings indicate that the goal-sensitive backbone consistently enhances performance in terms of both success rate and precision across all environments. Remarkably, in the **Flat** biome, our approach with the goal-sensitive backbone attains a 26% and 22% performance improvement in success rate and precision, respectively. This demonstrates that the goal-sensitive backbone effectively fuses the goal information into visual features and leads to goal-aware behavior.

Table 3. Additional ablation experiments on **Plains** biome.

#	Method	Avg. SR (%)	Avg. P (%)
1	Ours (GSB + horizon pred)	$\mathbf{55_{\pm09}}$	$\mathbf{70_{\pm09}}$
2	Ours + RNN	$\mathbf{65_{\pm07}}$	$\mathbf{67_{\pm08}}$
3	Ours − horizon pred + RNN	$39_{\pm08}$	$51_{\pm08}$
4	Ours − horizon pred	$35_{\pm08}$	$45_{\pm15}$
5	w/o horizon loss	$47_{\pm06}$	$54_{\pm08}$
6	w/o extra obs	$50_{\pm07}$	$69_{\pm07}$
7	w/o language condition	$25_{\pm03}$	$26_{\pm05}$

Table 4. The success rate (SR) under condition-free policy.

Goal					Avg.
Success Rate (%)	$44_{\pm19}$	$24_{\pm06}$	$23_{\pm11}$	$11_{\pm07}$	$25_{\pm03}$

Parameter sensitivity on horizon prediction. To investigate the sensitivity of the horizon-based control policy to the constant c (outlined in §3.3), we perform experiments with c values ranging from 0 to 14. We train and evaluate the model using the multi-task setting on the **Flat** benchmark, shown in Figure 5. Our findings indicate that within the 0 to 10 range, decreasing c enhances performance, while further reduction leads to decline. This implies that subtracting a small constant from the predicted horizon-to-goal yields a more effective policy. However, subtracting a larger value results in performance deterioration, as attaining the goal within such a limited horizon may be unfeasible.

Comparision with recurrent architecture. We built two recurrent variants ("Ours + RNN", "Ours − horizon pred + RNN") by using a GRU module to fuse the joint representation f_t and optionally also removing the horizon prediction module. During training, the batch size, frame number, and skipping frame are set to 8, 16, and 5, respectively. Table 3 (exp1 *vs.* exp3) shows that "Ours − horizon pred + RNN" becomes significantly worse, likely due to the partial observability issue (-26% SR). However, when combining RNN and horizon module (exp2), the performance gains significantly more than our original method ($+10\%$ SR). To sum up, while RNNs can aid in addressing partial observability, our findings indicate that in our open-world scenario, they are considerably more effective when combined with our horizon prediction module.

Ablation on horizon loss, extra observation, and language condition. Table 3 demonstrates that excluding horizon loss (exp5) and extra observation (exp6) can result in a decrease of success rate by 8% and 5%, respectively. Furthermore, as depicted in Table 4, when the language condition is removed from the input (exp7), the policy primarily accomplishes the "chopping tree" task (44% SR) while scarcely completing the "hunting pig" task (11% SR). The tasks "hunting sheep" and "hunting cow" are executed fairly evenly (around 24% SR). This is likely due to trees appearing more frequently than animals in the environment.
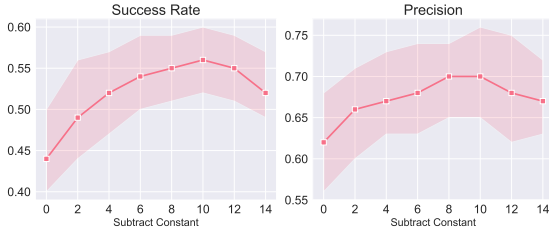
Figure 5. Multi-task performance as a function of subtracting the horizon constant c. Results show that setting c to a small constant lead to better overall performance as it incentivizes the agent to exhibit behaviors that lead to faster task completion.

4.4. Generalization Performance

In the open-ended Minecraft environment, which features a variety of biomes with distinct appearances, a decent agent should be capable of generalizing across these diverse biomes. To evaluate the agent's zero-shot generalization ability in a new biome, we initially train the agent using data exclusively from the Plains biome. Subsequently, we test it in the Flat biome, where it faces the challenge of combatting `sheep`, `cows`, and `pigs`. Complicating the task, numerous distracting mobs, such as `wolves` and `mushroom cows`, appear in the testing biome but not in the training biome. The results are presented in Table 5. Our zero-shot agent demonstrates success rates comparable to those of an agent trained directly on the Flat biome. The high precision of our zero-shot agent also indicates its robust performance, even amidst numerous novel distracting mobs in the new testing biome. Therefore, we believe that our agent displays a degree of zero-shot generalization to new environments, achieved through goal-aware representation learning and adaptive horizon prediction.

5. Related Works

Open-ended Environments. A variety of environments have been developed for open-ended agent training, such as grid worlds [8, 9], maze worlds [25, 42, 46], and indoor worlds [1, 15, 38, 40]. Although these benchmarks have advanced agent development, they generally lack complexity in perception and task domains. This paper concentrates on Minecraft, a voxel-based 3D, first-person, open-world game centered around survival and creation. Microsoft introduced the first Gym-style API platform called Malmo [24] for Minecraft, which has spawned numerous secondary development variants. Building on Malmo, MineRL [20] offers a human-interface simulator and a dataset of human play demonstrations for the annual Diamond Challenge at NeurIPS [18, 19, 26]. MineDoJo [16], an extension of MineRL, broadens the APIs for customizing tasks and provides thousands of pre-defined compositional tasks aimed at developing a generally capable embodied agent, which we use to evaluate our method.

Table 5. Quantitive results on generalization to a novel biome.

Train → Eval	Success Rate (%)				Precision (%)			
	🐑	🐄	🐖	Avg.	🐑	🐄	🐖	Avg.
Flat→Flat	72	60	57	**63**	44	48	54	**49**
Plains→Flat	67	47	60	**58**	89	89	70	**83**

Embodied Agents in Minecraft. Some prior studies have utilized a hierarchical reinforcement learning framework to develop sophisticated embodied agents. For instance, SEIHAI [31] divides a long-horizon task into several subtasks, training an appropriate agent for each subtask and designing a scheduler to manage the execution of these agents. Similarly, JueWu-MC [28] adopts this concept but enhances the agent with action-aware representation learning capabilities. In recent times, the internet-scale pretraining paradigm has made a significant impact on embodied research in open-ended environments. VPT [4], for example, undergoes pretraining on an extensive collection of online gameplay videos using imitation learning. However, it lacks the ability to process any command input. MineAgent [16] takes a different approach by pretraining a language-conditioned reward function using online video-transcript pairs, which is then utilized to support multi-task reinforcement learning.

Progress Monitor. The horizon-to-goal prediction technology has already been employed as a progress monitor in the Vision-Language Navigation (VLN) communities [29, 30, 47]. This technology aids in understanding the task structure and expediting the training procedure. Generally, current progress monitors primarily function as supplementary objectives. Their estimated progress is utilized to reassess actions or execute beam search. In contrast, our estimated horizon is explicitly incorporated into the policy network to guide agent behaviors. During inference, the horizon input can be adjusted for enhanced performance.

6. Conclusion

In this paper, we explore the issue of learning goal-oriented policies in open-world environments. We pinpoint two major challenges unique to such settings: 1) the difficulty in distinguishing tasks from the state distribution due to immense scene variety, and 2) the non-stationary nature of environmental dynamics resulting from partial observability. We propose a goal-sensitive backbone and an adaptive horizon prediction module to overcome both. Our experiments on challenging Minecraft confirm the advantages of our proposed methods over baselines in terms of both success rate and precision of task completeness.

References

[1] Josh Abramson, Arun Ahuja, Arthur Brussee, Federico Carnevale, Mary Cassin, Stephen R. Clark, Andrew Dudzik, Petko Georgiev, Aurelia Guy, Tim Harley, Felix Hill, Alden Hung, Zachary Kenton, Jessica Landon, Timothy P. Lillicrap, Kory Wallace Mathewson, Alistair Muldal, Adam Santoro, Nikolay Savinov, Vikrant Varma, Greg Wayne, Nathaniel Wong, Chen Yan, and Rui Zhu. Imitating interactive intelligence. *arXiv: Learning*, 2020. 8

[2] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. 1, 2

[3] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*, 2022. 1

[4] Bowen Baker, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (VPT): learning to act by watching unlabeled online videos. *CoRR*, abs/2206.11795, 2022. 3, 6, 7, 8, 12

[5] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. 4

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 1

[7] Han Cai, Chuang Gan, Ligeng Zhu, and Song Han. Tinytl: Reduce memory, not parameters for efficient on-device learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 11285–11297. Curran Associates, Inc., 2020. 12

[8] Tianshi Cao, Jingkang Wang, Yining Zhang, and Sivabalan Manivasagam. Babyai++: Towards grounded-language learning beyond memorization. *CoRR*, abs/2004.07200, 2020. 8

[9] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. *Learning*, 2018. 8

[10] Yiming Ding, Carlos Florensa, Pieter Abbeel, and Mariano Phielipp. Goal-conditioned imitation learning. *Advances in neural information processing systems*, 32, 2019. 2

[11] Shibhansh Dohare, A Rupam Mahmood, and Richard S Sutton. Continual backprop: Stochastic gradient descent with persistent randomness. *arXiv preprint arXiv:2108.06325*, 2021. 6

[12] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021. 3

[13] Islam Elnabarawy, Kristijana Arroyo, and Donald C. Wunsch. Starcraft ii build order optimization using deep reinforcement learning and monte-carlo tree search. *arXiv: Learning*, 2020. 1

[14] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *ICML*, pages 1407–1416. PMLR, 2018. 4, 6, 7

[15] Linxi Fan, Guanzhi Wang, De-An Huang, Zhiding Yu, Li Fei-Fei, Yuke Zhu, and Animashree Anandkumar. Secant: Self-expert cloning for zero-shot generalization of visual policies. *arXiv: Learning*, 2021. 8

[16] 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. *arXiv preprint arXiv:2206.08853*, 2022. 1, 2, 6, 7, 8, 12

[17] Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Manon Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. In *International Conference on Learning Representations*, 2021. 5

[18] William H. Guss, Mario Ynocente Castro, Sam Devlin, Brandon Houghton, Noboru Sean Kuno, Crissman Loomis, Stephanie Milani, Sharada P. Mohanty, Keisuke Nakata, Ruslan Salakhutdinov, John Schulman, Shinya Shiroshita, Nicholay Topin, Avinash Ummadisingu, and Oriol Vinyals. The minerl 2020 competition on sample efficient reinforcement learning using human priors. *arXiv: Learning*, 2021. 1, 8

[19] William H Guss, Cayden Codel, Katja Hofmann, Brandon Houghton, Noboru Kuno, Stephanie Milani, Sharada Mohanty, Diego Perez Liebana, Ruslan Salakhutdinov, Nicholay Topin, et al. Neurips 2019 competition: the minerl competition on sample efficient reinforcement learning using human priors. *arXiv preprint arXiv:1904.10079*, 2019. 1, 8

[20] William H. Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations. *international joint conference on artificial intelligence*, 2019. 1, 8

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4

[22] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022. 4

[23] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge J. Belongie, Bharath Hariharan, and Ser-Nam Lim.

Visual prompt tuning. In Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXXIII*, volume 13693 of *Lecture Notes in Computer Science*, pages 709–727. Springer, 2022. 12

[24] Matthew Johnson, Katja Hofmann, Tim J. Hutton, and David Michael Bignell. The malmo platform for artificial intelligence experimentation. *international joint conference on artificial intelligence*, 2016. 1, 2, 8

[25] Arthur Juliani, Ahmed Khalifa, Vincent Pierre Berges, Jonathan Harper, Ervin Teng, Hunter Henry, Adam Crespi, Julian Togelius, and Danny Lange. Obstacle tower: A generalization challenge in vision, control, and planning. *international joint conference on artificial intelligence*, 2019. 8

[26] Anssi Kanervisto, Stephanie Milani, Karolis Ramanauskas, Nicholay Topin, Zichuan Lin, Junyou Li, Jianing Shi, Deheng Ye, Qiang Fu, Wei Yang, Weijun Hong, Zhongyue Huang, Haicheng Chen, Guangjun Zeng, Yue Lin, Vincent Micheli, Eloi Alonso, Fran
c{c}ois Fleuret, Alexander Nikulin, Yury Belousov, Oleg Svidchenko, and Aleksei Shpilman. Minerl diamond 2021 competition: Overview, results, and lessons learned. *neural information processing systems*, 2022. 1, 8

[27] Jin-Hwa Kim, Kyoung-Woon On, Woosang Lim, Jeonghee Kim, Jung-Woo Ha, and Byoung-Tak Zhang. Hadamard product for low-rank bilinear pooling. *arXiv preprint arXiv:1610.04325*, 2016. 4

[28] Zichuan Lin, Junyou Li, Jianing Shi, Deheng Ye, Qiang Fu, and Wei Yang. Juewu-mc: Playing minecraft with sample-efficient hierarchical reinforcement learning. *arXiv preprint arXiv:2112.04907*, 2021. 8

[29] Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. Self-monitoring navigation agent via auxiliary progress estimation. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. 8

[30] Chih-Yao Ma, Zuxuan Wu, Ghassan AlRegib, Caiming Xiong, and Zsolt Kira. The regretful agent: Heuristic-aided navigation through progress estimation. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 6732–6740, 2019. 8

[31] Hangyu Mao, Chao Wang, Xiaotian Hao, Yihuan Mao, Yiming Lu, Chengjie Wu, Jianye Hao, Dong Li, and Pingzhong Tang. Seihai: A sample-efficient hierarchical ai for the minerl competition. In *International Conference on Distributed Artificial Intelligence*, pages 38–51. Springer, 2021. 8

[32] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv: Learning*, 2013. 1, 2

[33] Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. In *International Conference on Machine Learning*, pages 2661–2670. PMLR, 2017. 4

[34] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 4

[35] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022. 1

[36] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied ai research. *international conference on computer vision*, 2019. 2

[37] Juergen Schmidhuber. Reinforcement learning upside down: Don't predict rewards–just map them to actions. *arXiv preprint arXiv:1912.02875*, 2019. 5

[38] Bokui Shen, Fei Xia, Chengshu Li, Roberto Martín-Martín, Linxi Fan, Guanzhi Wang, Shyamal Buch, Claudia D'Arpino, Sanjana Srivastava, Lyne P. Tchapmi, Micael Tchapmi, Kent Vainio, Li Fei-Fei, and Silvio Savarese. igibson, a simulation environment for interactive tasks in large realistic scenes. *intelligent robots and systems*, 2020. 8

[39] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017. 1, 2

[40] Sanjana Srivastava, Chengshu Li, Michael Lingelbach, Roberto Martín-Martín, Fei Xia, Kent Vainio, Zheng Lian, Cem Gokmen, Shyamal Buch, C. Karen Liu, Silvio Savarese, Hyowon Gweon, Jiajun Wu, and Li Fei-Fei. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. *Conference on Robot Learning*, 2021. 8

[41] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018. 2

[42] Open Ended Learning Team, Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michaël Mathieu, Nat McAleese, Nathalie Bradley-Schmieg, Nathaniel Wong, Nicolas Porcel, Roberta Raileanu, Steph Hughes-Fitt, Valentin Dalibard, and Wojciech Marian Czarnecki. Open-ended learning leads to generally capable agents. *CoRR*, abs/2107.12808, 2021. 8

[43] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. *intelligent robots and systems*, 2012. 2

[44] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019. 1

[45] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John P. Agapiou, Julian Schrittwieser, John Quan, Stephen Gaffney, Stig Petersen, Karen

Simonyan, Tom Schaul, Hado van Hasselt, David Silver, Timothy P. Lillicrap, Kevin Calderone, Paul Keet, Anthony Brunasso, David Lawrence, Anders Ekermo, Jacob Repp, and Rodney Tsing. Starcraft ii: A new challenge for reinforcement learning. *arXiv: Learning*, 2017. 1

[46] Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O. Stanley. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv: Neural and Evolutionary Computing*, 2019. 8

[47] Joel Ye, Dhruv Batra, Abhishek Das, and Erik Wijmans. Auxiliary tasks and exploration enable objectnav. *arXiv preprint arXiv:2104.04112*, 2021. 8

[48] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. 2019. 2

[49] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020. 1, 13

[50] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *CoRR*, abs/2106.10199, 2021. 12

A. Experimental Details

A.1. Observation and Action Space

The agent receives identical information as human players do. The observation space primarily comprises four components: 1) ego-centric RGB frames, 2) voxels (surrounding blocks), 3) GPS locations (the agent's three-dimensional coordinates), and 4) compass (pitch/yaw angles). These are shaped as $(3, 480, 640)$, $(3, 3, 3)$, $(3,)$, and $(2,)$, respectively. It is important to note that the agent **does not know** the precise location of the target object. Instead, the agent can only obtain information about the target object by examining the pixel image. The RGB frames are resized to a shape of $(3, 128, 128)$ using bilinear interpolation before being fed into the networks. At each step, the agent must execute a movement action, camera action, and functional action. A compound action space is employed, consisting of a multi-discrete space with six dimensions: 1) forward and backward, 2) move left and right, 3) jump, sneak and sprint, 4) camera delta pitch, 5) camera delta yaw, and 6) functional actions (attack and use). The original delta camera degree, which ranges from -180 to 180, is discretized into 11 bins. As this paper's primary focus is on resource collection rather than item crafting, actions related to crafting are omitted.

A.2. Data Collection Pipeline

Our data collection pipeline collects high-quality goal-conditioned demonstrations with actions. The core idea is to train a proxy policy with non-goal demonstrations and roll out in customized environments, then filter the demonstrations according to the achievement. Generally, the pipeline consists of six steps: 1) collect online videos, 2) clean and label the videos, 3) train a proxy policy, 4) customize the environments, 5) roll out the proxy policy, and 6) filter by the accomplishments.

Video-Pretraining [4] is ideally suited for stages 1-3. It begins by amassing a vast dataset of Minecraft videos, sourced from the web using relevant keywords. Given that collected videos often feature overlaid artifacts, the process filters out videos without visual artifacts and those from survival mode. Next, an Inverse Dynamics Model (IDM) is trained to label these videos with actions, yielding demonstrations for proxy policy training. We directly employ the pretrained VPT[4] as our proxy policy. In stage 4, we utilize APIs supplied by MineDojo[16] to create environments tailored to each task's success criteria. During stage 5, we deploy the proxy policy, recording successful trajectories and their corresponding achieved goals. The environment is reset once the episode concludes or the goal is accomplished, ensuring trajectory independence.

Notably, we execute the proxy policy rollout in parallel using 16 processes on 4 A40 GPUs, generating 0.5GB of demonstrations per minute (without leveraging video compression algorithm during storing frames). This approach minimizes human intervention and enhances data collection efficiency. In total, we have gathered 215GB, 289GB, and 446GB of goal-conditioned demonstrations from **Plains**, **Flat**, and **Forest** environments, respectively.

A.3. Implementation

Horizon discretization. As the horizon illustrates the number of steps required to attain the desired objective, it is infeasible to precisely determine the exact value. In practice, we suggest dividing the original horizon into 16 distinct segments: $[0, 10) \rightarrow 0$, $[10, 20) \rightarrow 1$, $[20, 30) \rightarrow 2$, $\cdots$, $[90, 100) \rightarrow 9$, $[100, 120) \rightarrow 10$, $[120, 140) \rightarrow 11$, $\cdots$, $[180, 200) \rightarrow 14$, and $[200, \infty) \rightarrow 15$. In this approach, each segment inherently represents a phase that signifies the level of task completion. Consequently, the horizon prediction issue can be framed as a multi-class problem. It is important to note that the method of discretization is not singular and merits further exploration in the future.

Training. The observation of RGB image is scaled into 128×128 where no data augmentation is adopted. We train the policy with the AdamW optimizer and a linear learning rate decay. We use an initial learning rate of 0.0001, a batch size of 32, and a weight decay of 0.0001. Besides, we also use a warmup trick that the learning rate linearly increases from 0 to 0.0001 in 10k iterations. The policy is trained for 500k iterations on our collected dataset. It takes one day on a single A40 GPU. To train the baseline policies BC (VPT) and BC (CLIP), we only finetune the bias terms of their backbones, which is widely adopted by previous works [7, 23, 50]. Also note that, to keep the architecture comparable, we only transfer model and weights of the backbone from vpt model and MineCLIP model while replace their transformer architecture with ours.

Evaluation. During the evaluation, the maximum episode length is empirically set to 600, 600, and 300 for the **Flat**, **Plains**, and **Forest** biomes, respectively. In most instances, the agent is able to complete the assigned tasks within these limits. Furthermore, in our adaptive horizon prediction module, the hyperparameter c is empirically set to 3. The model is evaluated every 10,000 gradient updates. During each evaluation round, each goal is assessed 10 times to compute the *Success Rate* and *Precision* metrics. For the ablation study, we utilize the checkpoint after 500,000 training iterations, evaluate each goal 200 times, and report the average metrics in Table 5 and Figure 5.

B. Horizon Distribution Analysis

To further emphasize the importance of our adaptive horizon prediction module, we have visualized the distribution of successful trajectory lengths for various tasks in Minecraft, as shown in Figure 6. These successful trajec-
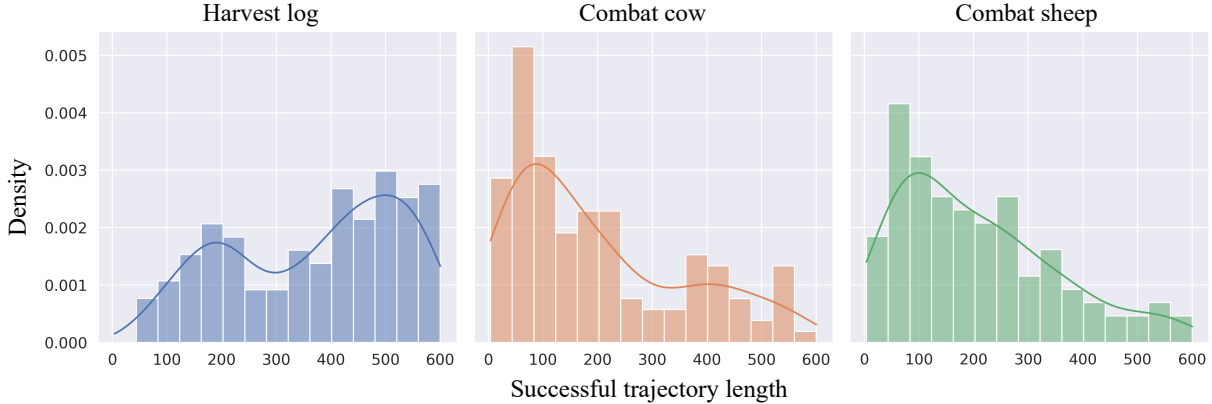
Figure 6. **Successful trajectory distribution of different tasks in open-ended Minecraft**. The distribution is long-tailed, making it hard to learn goal-conditioned policies with a fixed horizon.

tories were gathered from agents trained using single-task behavior cloning (with a randomly initialized Impala CNN as the backbone) in the `Plains` biome.

As depicted in Figure 6, the distribution of successful trajectory lengths in the open-world setting exhibits a long tail, making it challenging to train a policy with a fixed horizon. This can be attributed to Minecraft's extensive explorable space, partial observation properties, and non-stationary dynamics, which set it apart from other popular multi-task, closed-ended environments like Meta-World [49].

Consequently, the minimum number of steps needed for an agent to achieve its goal varies across different environments and episodes. The episode length typically hinges on the relative position and terrain constraints between the target object and the agent's initial position. An added layer of complexity arises when no target objects are near the agent's starting location, necessitating large-scale exploration (i.e., a larger horizon). Once the agent locates the target object, it must track it until the relevant skill can be executed on the object (e.g., killing or harvesting). This demands that the agent remain aware of its current stage.

Our proposed adaptive horizon prediction module incorporates the horizon as an additional condition for the policy. The policy explicitly takes into account the remaining time steps needed to achieve specific goals. Our experiments in Section 4.3 demonstrate that the adaptive horizon prediction module and the horizon loss $\mathcal{L}_h$ effectively enhance the success rate in open-world environments with such distributions.

C. Limitation and Future Work

In essence, our approach hinges on trajectories labeled with goals, which enables it to generalize across various domains, provided that such data is accessible. When only video segments labeled with actions are available, we can employ a goal predictor to assign goal labels to these clips. This can also be achieved by utilizing zero-shot models, such as CLIP. Moreover, if action labels are absent in these clips, we can resort to training an inverse dynamics model, as demonstrated in VPT. Undoubtedly, these present intriguing avenues for future exploration