JueWu-MC: Playing Minecraft with Sample-efficient Hierarchical Reinforcement Learning

Zichuan Lin *, Junyou Li *, Jianing Shi *, Deheng Ye, Qiang Fu and Wei Yang

Tencent AI Lab, Shenzhen, China
{zichuanlin, junyouli, jianingshi, dericye, leonfu, willyang}@tencent.com

Abstract
Learning rational behaviors in open-world games like Minecraft remains to be challenging for Reinforcement Learning (RL) research due to the compound challenge of partial observability, high-dimensional visual perception and delayed reward. To address this, we propose JueWu-MC, a sample-efficient hierarchical RL approach equipped with representation learning and imitation learning to deal with perception and exploration. Specifically, our approach includes two levels of hierarchy, where the high-level controller learns a policy to control over options and the low-level workers learn to solve each sub-task. To boost the learning of sub-tasks, we propose a combination of techniques including 1) action-aware representation learning which captures underlying relations between action and representation, 2) discriminator-based self-imitation learning for efficient exploration, and 3) ensemble behavior cloning with consistency filtering for policy robustness. Extensive experiments show that JueWu-MC significantly improves sample efficiency and outperforms a set of baselines by a large margin. Notably, we won the championship of the NeurIPS MineRL 2021 research competition and achieved the highest performance score ever.

1 Introduction
Deep reinforcement learning (DRL) has shown great success in many genres of games, including board game [Silver et al., 2016], Atari [Mnih et al., 2013], simple first-person-shooter (FPS) [Huang et al., 2019], real-time strategy (RTS) [Vinyals et al., 2019], multiplayer online battle arena (MOBA) [Berner et al., 2019; Ye et al., 2020b; Chen et al., 2021], etc. Recently, open-world games have been attracting attention due to its playing mechanism and similarity to real-world control tasks [Guss et al., 2021].

Minecraft, as a typical open-world game, has been increasingly explored for the past few years [Oh et al., 2016; Tessler et al., 2017; Guss et al., 2019; Kanervisto et al., 2020; Skrynnik et al., 2021; Mao et al., 2021].

Compared to other games, the characteristics of Minecraft make it a suitable testbed for RL research, as it emphasizes exploration, perception and construction in a 3D open world [Oh et al., 2016]. The agent is only provided with partial observability and occlusions. The tasks in the game are chained and long-term. Generally, human can make rational decisions to explore basic items and construct desired higher-level items using a reasonable amount of samples, while it can be hard for an AI agent to do so autonomously. Therefore, to facilitate the efficient decision-making of agents in playing Minecraft, MineRL [Guss et al., 2019] has been developed as a research competition platform, which provides human demonstrations and encourages the development of sample-efficient RL agents for playing Minecraft. Since the release of MineRL, a number of efforts have been made on developing Minecraft AI agents [Skrynnik et al., 2021; Mao et al., 2021].

However, it is still difficult for existing RL algorithms to mine items in Minecraft due to the compound challenge it poses, expanded below.

Long-time Horizons. In order to achieve goals (e.g., mining a diamond) in Minecraft, the agent is required to finish a variety of sub-tasks (e.g., log, craft) that highly depend on each other. Due to the sparse reward, it is hard for agents to learn long-horizon decisions efficiently. Hierarchical RL from demonstrations [Le et al., 2018; Pertsch et al., 2020] has been explored to leverage the task structure to accelerate the learning process. However, learning from unstructured demonstrations without domain knowledge remains challenging.

High-dimensional Visual Perception. Minecraft is a flexible 3D first-person game revolving around gathering resources (i.e., explore) and creating structures and items (i.e., construct). In this environment, agents are required to deal with high-dimensional visual input to enable efficient control. However, agent’s surroundings are varied and dynamic, which poses difficulties to learning a good representation.

Inefficient Exploration. With partial observability, the agent needs to explore in the right way and collect information from the environment so as to achieve goals. A naïve exploration strategy can waste a lot of samples on useless exploration. Self-imitation Learning (SIL) [Oh et al., 2018] is a simple method that learns to reproduce past good behaviors to incentivize deep exploration. However, SIL is not sample-efficient because its advantage-clipping operation causes a waste of
samples. Moreover, SIL does not make use of the transitions between samples.

**Imperfect Demonstrations.** Human demonstrations in playing Minecraft are highly distributional diverse [Kanervisto et al., 2020]. Also, there exists noisy data due to the imperfection of human operation [Guss et al., 2019].

To address the aforementioned compound challenges, we develop an efficient hierarchical RL approach equipped with novel representation and imitation learning techniques. Our method makes effective use of human demonstrations to boost the learning of agents and enables the RL algorithm to learn rational behaviors with high sample efficiency.

**Hierarchical Planing with Prior.** We first propose a hierarchical RL (HRL) framework with two levels of hierarchy, where the high-level controller automatically extracts sub-goals in long-horizon trajectories from the unstructured human demonstrations and learns a policy to control over options, while the low-level workers learn sub-tasks to achieve sub-goals by leveraging both demonstrations dispatched by the high-level controller and interactions with environments. Our approach automatically structures the demonstrations and learns a hierarchical agent, which enables better decision on long-horizon tasks. Under our HRL framework, we devise the following key techniques to boost agent learning.

**Action-aware Representation Learning.** Although some prior works [Huang et al., 2019] proposed using auxiliary tasks (e.g., enemy detection) to better understand the 3D world, such methods require a large amount of labeled data. We propose a self-supervised action-aware representation learning (A2RL) technique, which learns to capture the underlying relations between action and representation in 3D visual environments like Minecraft. As we will show, A2RL not only enables effective control by learning a compact representation but also improves the interpretability of the learned policy.

**Discriminator-based Self-imitation Learning.** As mentioned, existing self-imitation learning is advantage-based and becomes sample-inefficient for handling tasks in Minecraft, as it wastes a lot of samples due to the clipped objective and does not utilize transitions between samples. Therefore, we propose discriminator-based self-imitation learning (DSIL) which leverages self-generated experiences to learn self-correctable policies for better exploration.

**Ensemble Behavior Cloning with Consistency Filtering.** Learning a robust policy from imperfect demonstrations is difficult [Wu et al., 2019]. To address this issue, we first propose consistency filtering to identify the most common human behavior, and then perform ensemble behavior cloning to learn a robust agent with reduced uncertainty.

In summary, our contributions are: 1) We propose JouWu-MC, a sample-efficient hierarchical RL approach, equipped with novel techniques including action-aware representation learning, discriminator-based self-imitation, and ensemble behavior cloning with consistency filtering, for training Minecraft AI agents. 2) Our approach outperforms competitive baselines by a significantly large margin and achieves the best performance ever throughout the MineRL competition history. Thorough ablations and visualizations are further conducted to help understand why our approach works.

## 2 Related Work

**Game AI.** Game has long been a preferable field for artificial intelligence research. AlphaGo [Silver et al., 2016] mastered the game of Go with DRL and tree search. Since then, DRL has been used in other more sophisticated games, including StarCraft (RTS) [Vinyals et al., 2019], Google Football (Sports) [Huang et al., 2021], VizDoom (FPS) [Huang et al., 2019], Dota (MOBA) [Berner et al., 2019], Honor of Kings (MOBA) [Ye et al., 2020c; Ye et al., 2020a]. Recently, the 3D open-world game Minecraft is drawing rising attention. [Oh et al., 2016] showed that existing RL algorithms suffer from generalization in Minecraft and proposed a new memory-based DRL architecture. [Tessler et al., 2017] proposed H-DRLN, a combination of a deep skill array and a skill distillation system, to promote lifelong learning and transfer knowledge among different tasks in Minecraft. Since MineRL was held in 2019, many solutions have been proposed to learn to play in Minecraft. There works can be grouped into two categories: 1) end-to-end learning [Amiranashvili et al., 2020; Kanervisto et al., 2020]; 2) HRL with human demonstrations [Skrynnik et al., 2021; Mao et al., 2021]. Our approach belongs to the second category. In this category, prior works leverage the structure of the tasks and learn a hierarchical agent to play in Minecraft — ForgER [Skrynnik et al., 2021] proposed a hierarchical method with forgetful experience replay to allow the agent to learn from low-quality demonstrations; [Mao et al., 2021] proposed SEIHAI that fully takes advantage of the demonstrations and the task structure.

**Sample-efficient Reinforcement Learning.** Our work is to build a sample-efficient RL agent for playing Minecraft, and we thereby develop a combination of efficient learning techniques. We discuss the most relevant works below.

Our work is related to recent HRL research that builds upon human priors. To expand, [Le et al., 2018] proposed to warm-up the hierarchical agent from demonstrations and fine-tune with RL algorithms. [Pertsch et al., 2020] proposed to learn a skill prior from demonstrations to accelerate HRL algorithms. Compared to existing works, we are faced with the highly unstructured demo in 3D first-person video games played by the crowds. We address this challenge by structuring the demonstrations and defining sub-tasks and sub-goals automatically.

Representation learning in RL has two broad directions: self-supervised learning and contrastive learning. The former [Wu et al., 2021] aims at learning rich representations for high-dimensional unlabeled data to be useful across tasks, while the latter [Srinivas et al., 2020] learns representations that obey similarity constraints in a dataset organized by similar and dissimilar pairs. Our work proposes a novel self-supervised representation learning method that can measure action effects in 3D video games.

Existing methods use curiosity or uncertainty as a signal for exploration [Pathak et al., 2017] so that the learned agent is able to cover a large state space. However, the exploration-exploitation dilemma, given the sample efficiency consideration, drives us to develop self-imitation learning (SIL) [Oh et al., 2018] methods that focus on exploiting past good experiences for better exploration. Hence, we propose discriminator-
based self-imitation learning (DSIL) for efficient exploration.

Our work is also related to learning from imperfect demonstrations, such as DQfD [Hester et al., 2018] and Q-filter [Nair et al., 2018]. Most methods in this field leverage online interactions with the environment to handle the noise in demonstrations. We propose ensemble behavior cloning with consistency filtering (EBC) which leverages imperfect demonstrations to learn robust policies in playing Minecraft.

3 Method

In this section, we first introduce our overall HRL framework, and then illustrates the details of each component.

3.1 Overview

Figure 1 shows our overall framework. We define the human demonstrations as \( D = \{\tau_0, \tau_1, \tau_2, \ldots\} \) where \( \tau_i \) represents a long-horizon trajectory containing states, actions and rewards. The provided demonstrations are unstructured in that there are no explicit signals to specify sub-tasks and sub-goals.

We first define atomic skill as an individual skill that gets a non-zero reward. Then, we define sub-tasks and sub-goals based on atomic skill. To define reasonable sub-tasks, we examine the degree of reward delay for each atomic skill. We keep those atomic skills with long reward delay as individual sub-tasks because they require executing a long sequence of actions to achieve a delayed reward. Meanwhile, we merge those adjacent atomic skills with short reward delay into one sub-task. By doing so, we get \( n \) sub-tasks (a.k.a stages) in total. To define a sub-goal for each sub-task, we extract the most common human behavior pattern and use the last state in each sub-task as its sub-goal. In this way, we get structured demonstrations (\( D \rightarrow \{D_0, D_1, \ldots, D_{n-1}\} \)) with sub-tasks and sub-goals that are used to train the hierarchical agent. With the structured demonstrations, we train the meta-policy by imitation learning, and train the sub-policies to solve sub-tasks by leveraging both demonstrations and interactions with the environment, as described below.

3.2 Meta- and Sub-policies

Meta-policy. In principle, the meta-policy can be trained using RL algorithms. However, due to the strict constraints of training resources in MineRL, we resort to imitation learninging RL algorithms. However, due to the strict constraints of training resources in MineRL, we resort to imitation learning. To efficiently solve this kind of sub-tasks, we propose action-aware representation learning as well as discriminator-based self-imitation learning to facilitate the learning process of sub-policies. We show the model architecture in Figure 2. The full algorithm is shown in Appendix.

Action-aware Representation Learning

Learning compact representation is crucial to improve sample efficiency in reinforcement learning [Lesort et al., 2018]. To tackle the challenge of learning good representation in 3D open world, we start by observing that in first-person 3D environments, different actions have their own effects — each action acts on a local part of the high-dimensional observations. For example, in Minecraft, the attack action aims to break and acquire the block in front of the agent, while the camera action aims to adjust the agent’s camera perspective. Motivated by this observation, we propose action-aware representation learning (A2RL), to learn representation that can capture the underlying relation with actions.

To achieve so, we leverage the dynamic property from environments. Specifically, we learn a mask net on feature map for each action to capture dynamic information between the current and next states. Denote the feature map as...
to learn the underlying relations between representation and action by leveraging one-step dynamic prediction — this provides the agent with multi-view representations that reveal the effects of different actions. The learned representations can then be combined with any off-the-shelf RL algorithms to improve sample efficiency.

For policy-based methods, we plug our learned representations into policy \( \pi_\theta(a|1 + m_\phi(s, a)) \circ f_\theta(s) \) for effective perception and efficient back-propagation of policy gradient. For value-based methods, we combine our learned representation directly with Q-value functions \( Q_\pi_\theta((1 + m_\phi(s, a)) \circ f_\theta(s), a) \). The learning of Q-value function can be done using any Q-learning based algorithms.

**Discriminator-based Self-imitation Learning**

Self-imitation Learning (SIL) [Oh et al., 2018] is considered as a simple but effective way to solve hard-exploration tasks. SIL uses an advantage clipping technique to bias the agent towards good behaviors, which we call it as advantage-based self-imitation learning (ASIL). However, SIL is not sample-efficient due to the clipping mechanism. Besides, SIL does not leverage the transition between samples.

To address the issues of SIL, we propose discriminator-based self-imitation learning (DSIL). Unlike ASIL, DSIL does not use advantage clipping. Our intuition is that the agent should be encouraged to visit the state distribution that is more likely to lead to goals.

To do so, DSIL first learns a discriminator to distinguish between states from successful and failed trajectories (i.e., “good” and “bad” states), and then uses the learned discriminator to guide exploration. Specifically, we maintain two replay buffers \( B_+^\gamma \) and \( B_-^\gamma \) to store successful and failed trajectories respectively. During learning, we treat data from \( B_+^\gamma \) as positive samples and data from \( B_-^\gamma \) as negative samples to train the discriminator. Denote the discriminator as \( D_\xi : S \rightarrow [0, 1] \) which is parameterized by parameters \( \xi \). We train the discriminator with the following objective:

\[
\max_\xi \mathbb{E}_{s \sim B_+^\gamma} \left[ \log D_\xi(s) \right] + \mathbb{E}_{s \sim B_-^\gamma} \left[ 1 - \log D_\xi(s) \right].
\]

Intuitively, this objective encourages \( D_\xi(s) \) to output high values for good states while giving low values for bad states. For those states that are not distinguishable, \( D_\xi(s) \) tends to output 0.5. The learned discriminator captures the good state distribution that leads to goals and the bad state distribution that leads to failure.

We then use the trained discriminator to provide intrinsic rewards for policy learning to guide exploration. The intrinsic reward is defined as:

\[
\bar{r}(s, a, s') = \begin{cases} 
+1, & D_\xi(s') > 1 - \epsilon \\
-1, & D_\xi(s') < \epsilon 
\end{cases}
\]

where \( \epsilon \in (0, 0.5) \) is a hyper-parameter to control the confidence interval of \( D_\xi \). This reward drives the policy to explore in regions that previously get successful trajectories. By training the policy with this intrinsic reward, we encourage the policy to stay close to good state distribution so as to reproduce the agent’s past good decisions. Also, it is worth noting that DSIL encourages the policy to be self-correctable — it helps push the agent to the good state distribution even when the agent falls into the bad state distribution accidentally.
3.4 Learning Sub-policies to Craft Items

In this type of sub-task, agents must learn a sequence of actions to craft items. For example, to craft a wooden pickaxe, agents need to craft planks/sticks/tables, place tables, and finally craft a wooden pickaxe. In order to finish such tasks, agents need to learn a robust policy to execute a sequence of actions.

We explore pure imitation learning (IL) to reduce the need of interactions with the environment, due to the limited sample and interaction usage in MineRL. We propose ensemble behavior cloning with consistency filtering (EBC).

**Consistency Filtering.** Human demonstrations can be diverse and noisy. Directly imitating such noisy data can cause confusion for the policy since there might appear different expert actions under the same state. In order words, there might exist multiple modes of near-optimal behavior as well as sub-optimal behaviors in human demonstrations, and it can be difficult for BC to capture consistent good behaviors by learning from human demonstrations. Therefore, we perform consistency filtering by extracting the most common pattern of human behaviors. We show the algorithm in Appendix. We first extract the most common action sequence from demonstrations $D_i$ (line 1 to 10), and then keep those demonstrations that follow this action sequence while filtering the others, resulting in a smaller but consistent demonstration set $D_i$ (line 11 to 13).

**Ensemble Behavior Cloning.** Learning policy from offline demonstrations can suffer from generalization issues. Policy learned by BC will become uncertain when encountering unseen out-of-distribution states, which leads to erroneous output at each step and cause compounding errors.

To mitigate these issues, EBC learns a population of policies on different subsets of demonstrations to reduce the uncertainty of the agent’s decision. Specifically, we train $K$ policies on different demonstrations with NLL loss:

$$\min_{\theta_k} \mathbb{E}_{s, a \sim D_i^K} [- \log \pi_{\theta_k}(a|s)], \quad D_i^K \subset D_i, \quad k = 1, 2, \ldots, K, \quad (8)$$

where $\theta_k$ parameterizes the $k$-th policy. During inference, EBC adopts the majority voting mechanism to select an action that is the most confident among the population of policies. In this way, EBC learns a robust agent with reduced uncertainty and thus mitigates compounding errors.

4 Experiment

We conduct experiments on MineRL environment [Guss et al., 2019]. Our approach is built based on existing RL algorithms including SQIL [Reddy et al., 2019], PPO [Schulman et al., 2017], DQfD [Hester et al., 2018]. We present the details of our approach as well as the experiment settings in Appendix.

We aim to answer the following questions: (1) How does our approach perform compared with baselines in terms of sample efficiency and final performance? (2) How does each proposed techniques contribute to the overall performance? (3) How do the proposed techniques work and help improve sample efficiency during training?

**Baselines.** We compare our approach with several baselines in two categories: 1) end-to-end learning methods used in prior work and in the official implementation provided, including BC [Kanervisto et al., 2020], SQIL [Reddy et al., 2019], Rainbow [Hessel et al., 2018], DQfD [Hester et al., 2018], PDDQN [Schaul et al., 2015; Van Hasselt et al., 2016; Wang et al., 2016]; all these baselines are trained within 8 million samples with default hyper-parameters. 2) The top solutions from MineRL 2019&2020&2021 including SEIHAI [Mao et al., 2021] (1st place in MineRL 2020) and ForgER [Skrynnik et al., 2021] (1st place in MineRL 2019).

4.1 Main Results

Table 1 shows all the MineRL competition results since it was held in 2019. It is important to mention that the competition settings in MineRL 2020&2021 are much more difficult than MineRL 2019, e.g., the obfuscation in state and action space, making it impossible to work on action/state tricks. In this case, participants have to focus on the algorithm design itself in 2020 and 2021. Therefore, the scores in MineRL 2020&2021 are lower than those in MineRL 2019, which is explained in [Guss et al., 2021]. Overall, our approach outperforms all previous solutions, even including MineRL 2019, by a very large margin. We also find that end-to-end baselines cannot get a decent result, which indicates that it is quite challenging to solve such a long-horizon task with end-to-end learning. Compared with the results of MineRL 2020 (directly comparable with MineRL 2021 as they share the same settings), our method significantly outperforms other competitive solutions with a score (76.97) that is 3.4x higher compared to the second place (22.97), and even outperforms the sum of all other competitors’ scores (75.25) in MineRL 2020. We also compare the conditional success rate of each stage between our approach and SEIHAI in Table 2 and find that our method outperforms SEIHAI in every stages clearly.

We show the training curves in Figure 3(a). Note that due to the version update of MineRL 2021 (0.3.7 $\rightarrow$ 0.4.2) in online platform, many teams suffer from a performance drop.

<table>
<thead>
<tr>
<th>Baselines</th>
<th>2019 Competition Results</th>
<th>2020 Competition Results</th>
<th>2021 Competition Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Score</td>
<td>Team Name</td>
<td>Score</td>
</tr>
<tr>
<td>SQIL</td>
<td>2.94</td>
<td>CDS (ForgER)</td>
<td>61.61</td>
</tr>
<tr>
<td>DQfD</td>
<td>2.39</td>
<td>mc_rl</td>
<td>42.41</td>
</tr>
<tr>
<td>Rainbow</td>
<td>0.42</td>
<td>I4DS</td>
<td>40.8</td>
</tr>
<tr>
<td>PDDQN</td>
<td>0.11</td>
<td>CraftRL</td>
<td>23.81</td>
</tr>
<tr>
<td>BC</td>
<td>2.40</td>
<td>UEFDRL</td>
<td>17.9</td>
</tr>
</tbody>
</table>

Table 1: MineRL Competition Results. Our solution (JueWu-MC) significantly outperforms all other competitive solutions.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Stage</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEIHAI</td>
<td></td>
<td>64%</td>
<td>18.6%</td>
<td>78.5%</td>
<td>84.7%</td>
<td>23%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>JueWu-MC</td>
<td></td>
<td>92%</td>
<td>97%</td>
<td>98%</td>
<td>87%</td>
<td>46%</td>
<td>11%</td>
<td>0%</td>
</tr>
</tbody>
</table>

Table 2: The conditional success rate of each stage.
during online evaluation. Therefore, our online score (76.97) has a drop comparing with the performance in our training curve (>100). Even though, all the baselines we compare in the training curves share the same version (0.3.7) of MineRL, which is a fair comparison. Our approach is sample-efficient and outperforms the prior best results with only 0.5 million training samples. Our score reaches 100 with only 2.5 million training samples, which is much less than the limited number of samples (8 million) in MineRL competition. We do not include the training curve of ForgER [Skrynnik et al., 2021] because their environment setting (MineRL 2019) is easier than ours and thus is not comparable in sample efficiency.

4.2 Ablation Study

To examine the effectiveness of our proposed techniques, we consider three variants of our approach: 1) w/o A2RL, 2) w/o DSIL, and 3) w/o EBC. We show the training curves for each in Figure 3(b). We find that each proposed technique has a decent contribution to the overall performance. We also find that both EBC and A2RL contribute more than DSIL. This is because DSIL mainly boosts the performance for the later sub-tasks (i.e., stage 5 in appendix) while A2RL and EBC have earlier effects to the overall pipeline. EBC also contributes a lot to the overall performance, which demonstrates that learning a robust policy is quite important to solve a long-horizon task.

4.3 Visualization

To gain insights into why our learning techniques work, we conduct an in-depth analysis. First, to understand the learned mask in A2RL, we compute saliency maps for the action-aware mask generator. Specifically, to visualize the salient part of the images as seen by the mask net, we compute the absolute value of the Jacobian $|\nabla_s m_\phi(s, a)|$. Figure 4 shows the visualizations for three actions (attack, turn left, turn down). For each action $a$, we show current states, next states, and the saliency map on current states. We find that the learned mask is able to capture the dynamic information between two adjacent states, revealing the curiosity on the effect of actions. Specifically, the mask net learns to focus on uncertain parts of the current state, i.e., the parts that introduce uncertainty to the dynamic model. By focusing on the uncertain parts, the mask net learns to provide multiple views for the current state which can reflect the effects of different actions. For the ‘attack’ action, the learned mask attends to objects in front of the agent — this can help the agent pay more attention to the objects that will be attacked. For the ‘turn left’ and ‘turn down’ actions, the mask net learns to focus on the parts that have major change due to the rotation and translation of the agent’s perspective. Our learned mask assists the agent to better understand the 3D environment, which thus enables the agent to learn to control efficiently. Moreover, A2RL improves the interpretability of the learned policy.

To understand how DSIL works, we also visualize the state distribution that the agent visits. We compare ‘PPO’, ‘PPO+SIL’ and ‘PPO+DSIL’. For each method, we plot the visiting state distribution in different training stages. Figure 5 shows the visualization results. At the early training stage, both methods explore randomly and sometimes get to the goal state successfully. After getting enough samples and training for several epochs, ‘PPO+DSIL’ starts to explore in a more compact region, while ‘PPO’ and ‘PPO+SIL’ still explore in a wide region. This is because DSIL can push the agent to stay close to good state distribution, reproduce its past good behaviors and explore in the right way, which incentivizes the agent’s deep exploration to get successful trajectories.

5 Conclusion

In this paper, we present JueWu-MC, a sample-efficient hierarchical reinforcement learning framework designed to play Minecraft. With a high-level controller and several auto-extracted low-level workers, our framework can adapt to different environments and solve a series of sophisticated tasks. Furthermore, by introducing novel techniques on representation learning and imitation learning, we improve the performance and learning efficiency of the sub-policies. Experiments show that our pipeline significantly outperforms all the baseline algorithms and the solutions from past MineRL competitions. In future work, we would like to apply JueWu-MC to other Minecraft tasks, as well as other open-world games.
References


