JueWu-MC: Achieving Sample-Efficient Minecraft Gameplay through Hierarchical Reinforcement Learning

Abstract

Learning rational behaviors in open-world games such as Minecraft continues to pose a challenge to Reinforcement Learning (RL) research, due to the combined difficulties of partial observability, high-dimensional visual perception, and delayed rewards. To overcome these challenges, we propose JueWu-MC, a sample-efficient hierarchical RL method that incorporates representation learning and imitation learning to handle perception and exploration. Our approach has two levels of hierarchy: the high-level controller learns a policy to manage options, while the low-level workers learn to solve each sub-task. To boost learning of sub-tasks, we propose a combination of techniques including: 1) action-aware representation learning, which captures 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 demonstrate that JueWu-MC significantly enhances sample efficiency and outperforms several baselines. We won the championship of the MineRL 2021 research competition and achieved the highest performance score.

1 Introduction

Deep reinforcement learning (DRL) has achieved great success in numerous game genres including board games, Atari games, simple first-person-shooter (FPS) games, real-time strategy (RTS) games, and multiplayer online battle arena (MOBA) games. Recently, open-world games have garnered attention due to their playing mechanisms and their resemblance to real-world control tasks. Minecraft, as a typical open-world game, has been increasingly explored in recent years.

Compared to other games, the properties of Minecraft make it an ideal testbed for RL research, because it emphasizes exploration, perception, and construction in a 3D open world. Agents are given partial observability and face occlusions. Tasks in the game are chained and long-term. Humans can typically make rational decisions to explore basic items and construct more complex items with a reasonable amount of practice, while it can be challenging for AI agents to do so autonomously. To facilitate the effective decision-making of agents in playing Minecraft, MineRL 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 its release, many efforts have been made to develop Minecraft AI agents.

However, it remains difficult for current RL algorithms to acquire items in Minecraft due to several factors, which include the following. First, in order to reach goals, the agent is required to complete many sub-tasks that highly depend on each other. Due to the sparse reward, it is difficult for agents to learn long-horizon decisions efficiently. Hierarchical RL from demonstrations has been explored to take advantage of the task structure to accelerate learning. However, learning from unstructured demonstrations without any domain knowledge remains difficult. Second, Minecraft is a flexible 3D first-person game which revolves around gathering resources and creating structures and items. In this environment, agents are required to handle high-dimensional visual input to enable efficient

.

control. However, the agent's surroundings are varied and dynamic, which makes it difficult to learn a good representation. Third, with partial observability, the agent needs to explore in the right way and collect information from the environment in order to achieve goals. Naive exploration can waste a lot of samples on useless actions. Self-imitation learning (SIL) is a simple method that learns to reproduce past good behaviors to incentivize exploration, but it is not sample efficient. Lastly, human demonstrations are diverse and often noisy.

To address these combined challenges, we propose an efficient hierarchical RL approach, equipped with novel representation and imitation learning techniques. Our method leverages human demonstrations to boost the learning of agents, enabling the RL algorithm to learn rational behaviors with high sample efficiency.

Hierarchical Planning with Prior. We propose a hierarchical RL (HRL) framework with two levels of hierarchy. The high-level controller extracts sub-goals from human demonstrations and learns a policy to control options, while the low-level workers learn sub-tasks to achieve sub-goals by leveraging demonstrations and interactions with environments. Our approach structures the demonstrations and learns a hierarchical agent, which enables better decisions over long-horizon tasks. We use the following key techniques to boost agent learning.

Action-aware Representation Learning. We propose action-aware representation learning (A2RL) to capture the relations between action and representation in 3D visual environments such as Minecraft. A2RL enables effective control and improves the interpretability of the learned policy.

Discriminator-based Self-imitation Learning. 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. We propose consistency filtering to identify common human behaviors, and then perform ensemble behavior cloning to learn a robust agent with reduced uncertainty.

Our contributions are as follows: 1) We propose JueWu-MC, a sample-efficient hierarchical RL approach, equipped with action-aware representation learning, discriminator-based self-imitation, and ensemble behavior cloning with consistency filtering. 2) Our approach outperforms competitive baselines and achieves the best performance throughout the history of the competition.

2 Related Work

2.1 Game AI

Games have long been a testing ground for artificial intelligence research. AlphaGo mastered the game of Go with DRL and tree search. Since then, DRL has been used in other sophisticated games, including StarCraft, Google Football, VizDoom, and Dota. Recently, the 3D open-world game Minecraft has been attracting attention. Previous research has shown that existing RL algorithms can struggle to generalize in Minecraft and a new memory-based DRL architecture was proposed to address this. Another approach combines a deep skill array and a skill distillation system to promote lifelong learning and transfer knowledge among different tasks. Since the MineRL competition began in 2019, many solutions have been proposed to learn to play in Minecraft. These works can be grouped into two categories: 1) end-to-end learning; 2) hierarchical RL with human demonstrations. Our approach belongs to the second category, which leverages the structure of the tasks and learns a hierarchical agent to play in Minecraft. ForgER proposed a hierarchical method with forgetful experience replay, and SEIHAI fully takes advantage of human demonstrations and task structure.

2.2 Sample-efficient Reinforcement Learning

Our work aims to create 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 HRL research that builds upon human priors. One approach proposes to warm-up the hierarchical agent from demonstrations and fine-tune it with RL algorithms. Another approach proposes to learn a skill prior from demonstrations to accelerate HRL algorithms. Compared to existing works, we are faced with highly unstructured demos 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. Self-supervised learning aims to learn rich representations for high-dimensional unlabeled data to be useful across tasks. Contrastive learning learns representations that obey similarity constraints. Our work proposes a self-supervised representation learning method that measures action effects in 3D video games.

Existing methods use curiosity or uncertainty as a signal for exploration so that the learned agent is able to cover a large state space. The exploration-exploitation dilemma drives us to develop self-imitation learning (SIL) methods that focus on exploiting past good experiences for better exploration. We propose discriminator-based self-imitation learning (DSIL).

3 Method

In this section, we first introduce our overall HRL framework, and then describe each component in detail.

3.1 Overview

Our overall framework is shown in Figure 1. We define human demonstrations as $D = \{\tau_0, \tau_1, \tau_2, ...\}$ where τ_i is a long-horizon trajectory containing states, actions, and rewards. The provided demonstrations are unstructured, without explicit signals that specify sub-tasks and sub-goals.

We define an atomic skill as a skill that gets a non-zero reward. We define sub-tasks and sub-goals based on the atomic skills. To define sub-tasks, we examine the reward delay for each atomic skill, keeping those with long reward delays as individual sub-tasks and merging those with short reward delays into one sub-task. Through this process, we have n sub-tasks in total. To define sub-goals 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. Through this, we have structured demonstrations ($D \rightarrow \{D_0, D_1, ..., D_{n-1}\}$) with sub-tasks and sub-goals used to train the hierarchical agent. With the structured demonstrations, we train the meta-policy using imitation learning and train sub-policies to solve sub-tasks using demonstrations and interactions with the environment.

3.2 Meta- and Sub-policies

Meta-policy. We train a meta-policy that maps continuous states to discrete indices (0, 1, ..., n - 1) that specify which option to use. Given state space $\mathcal{S}$ and discrete option $o \in \mathcal{O}$, the meta-policy is defined as $\pi_m(\theta)(o|s)$, where $s \in \mathcal{S}$, $o \in \mathcal{O}$, and θ represents the parameters. $\pi_m(\theta)(o|s)$ specifies the conditional distribution over the discrete options. To train the meta-policy, we generate training data (s, i) where i represents the i-th stage and $s \in D_i$ is sampled from the demonstrations of the i-th stage. The meta-policy is trained using negative log-likelihood (NLL) loss:

$L_m = -\sum_{i=0}^{n-1} \log \pi_m(i|s)$

During inference, the meta-policy generates options by taking

$\sigma = argmax_o \pi_m(o|s)$

Sub-policy. In Minecraft, sub-tasks can be grouped into two main types: gathering resources, and crafting items. In the first type (gathering resources), agents need to navigate and gather sparse rewards by observing high-dimensional visual inputs. In the second type (crafting items), agents need to execute a sequence of actions robustly.

In typical HRL, the action space of the sub-policies is predefined. However, in the competition, a handcrafted action space is prohibited. Additionally, the action space is obfuscated in both human demonstrations and the environment. Learning directly in this continuous action space is challenging as exploration in a large continuous space can be inefficient. We use KMeans to cluster actions for each sub-task using demonstration D_i, and perform reinforcement learning and imitation learning based on the clustered action space.

In the following section, we describe how to learn sub-policies efficiently to solve these two kinds of sub-tasks.

3.3 Learning Sub-policies to Gather Resources

To efficiently solve these sub-tasks, we propose action-aware representation learning and discriminator-based self-imitation learning to facilitate the learning of sub-policies. The model architecture is shown in Figure 2.

Action-aware Representation Learning. To learn compact representations, we observe that in 3D environments, different actions have different effects on high-dimensional observations. We propose action-aware representation learning (A2RL) to capture the relation with actions.

We learn a mask net on a feature map for each action to capture dynamic information between the current and next states. Let the feature map be $f_\theta(s) \in \mathbb{R}^{C\times H\times W}$ and the mask net be $m_\phi(s,a) \in [0,1]^{H\times W}$, where θ and ϕ represent parameters of the policy and mask net. Given a transition tuple (s,a,s'), the loss function for training the mask is:

$$L_m(\phi) = -E_{s,a,s'\sim D}[\log(\sigma((f_\theta(s') - g_\psi(f_\theta(s))) \odot m_\phi(s,a))) + \eta(1 - m_\phi(s,a))]$$

where g_ψ is a linear projection function parameterized by learnable parameters ψ; $\odot$ represents element-wise product, and η is a hyper-parameter that balances two objectives.

To optimize the above loss function, we use a two-stage training process. In the first stage, we train the linear projection network g_{ψ_a} using the following objective:

$$L_g(\psi_a) = E_{s,a,s'\sim D}[||f_\theta(s') - g_{\psi_a}(f_\theta(s))||_2]$$

This objective learns to recover information of s' from s in latent space, which is equal to learning a dynamic model to predict the next state given the current state and action. Note that the parameter ψ is dependent on the action a. In the second stage, we fix the learned linear function g_{ψ_a} and optimize the mask net.

By minimizing the loss function, the mask net will learn to focus on local parts of the current image that are uncertain to the dynamic model. This is similar to human curiosity, which focuses on that which is uncertain.

For policy-based methods, we integrate our learned representations into policy networks. For value-based methods, we combine our learned representations directly with Q-value functions. The learning of the Q-value function can be done using any Q-learning based algorithms.

Discriminator-based Self-imitation Learning. 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 learns a discriminator to distinguish between states from successful and failed trajectories, and then uses the learned discriminator to guide exploration. We maintain two replay buffers B_i^+ and B_i^- to store successful and failed trajectories. During learning, we treat data from B_i^+ as positive samples and data from B_i^- as negative samples to train the discriminator. Let the discriminator be $D_\xi : S \to [0,1]$ which is parameterized by parameters ξ. We train the discriminator with the objective:

$$max_\xi E_{s\in B_i^+}[\log D_\xi(s)] + E_{s\in B_i^-}[1 - \log D_\xi(s)]$$

The discriminator is encouraged to output high values for good states and low values for bad states. For states that are not distinguishable, $D_\xi(s)$ tends to output 0.5.

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

$$r(s,a,s') = \{ +1 if D_\xi(s') > 1 - \epsilon$$
$$-1 if D_\xi(s') < \epsilon$$

where $\epsilon \in (0, 0.5)$ is a hyper-parameter to control the confidence interval of D_ξ. This reward drives the policy to explore in regions that previously led to successful trajectories. DSIL encourages the policy to stay close to a good state distribution, reproduce past decisions, and also be self-correctable.

3.4 Learning Sub-policies to Craft Items

In this type of sub-task, agents must learn a sequence of actions to craft items. 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 for interactions with the environment, due to the limited sample and interaction usage. 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. Therefore, we perform consistency filtering by extracting the most common pattern of human behaviors. We extract the most common action sequence from demonstrations D_i. For each trajectory, we keep those actions that lead to a state change while appearing for the first time to form an action sequence, and count the occurrences of each pattern. We then get the most common action pattern. Afterward, we conduct consistency filtering using the extracted action pattern.

Ensemble Behavior Cloning. Learning policy from offline datasets can lead to generalization issues. Policies learned through behavior cloning may become uncertain when encountering unseen out-of-distribution states. To mitigate this, 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:

$min_{\theta_k} E_{s,a\sim\bar{D_i^k}}[-\log \pi_{\theta_k}(a|s)]$, $\bar{D_i^k} \subset \bar{D}_i$, $k = 1, 2, ..., K$

where θ_k parameterizes the k-th policy. During inference, EBC adopts a majority voting mechanism to select an action that is the most confident among the policies.

4 Experiment

We conduct experiments using the MineRL environment. Our approach is built based on RL algorithms including SQIL, PPO, and DQfD.

4.1 Main Results

Table 1 shows all the MineRL competition results since 2019. The competition settings in 2020 and 2021 were more difficult than in 2019. In these years, participants had to focus on the algorithm design itself. The scores in 2020 and 2021 are lower than in 2019. Our approach outperforms all previous solutions. End-to-end baselines cannot achieve a decent result, showing it is difficult to solve long-horizon tasks with end-to-end learning. Compared to the results of the 2020 competition, our method outperforms other solutions with a score (76.97) that is 3.4x higher than the second place score (22.97). Table 2 shows the conditional success rate of each stage between our approach and SEIHAI. Our approach outperforms SEIHAI in every stage.

Figure 3(a) shows the training curves. Due to a version update of MineRL 2021, our online score dropped compared with the performance in our training curve. Our approach is sample-efficient and outperforms prior best results with 0.5 million training samples. Our score reaches 100 with 2.5 million training samples, which is less than the 8 million samples of the MineRL competition.

4.2 Ablation Study

To examine the effectiveness of our proposed techniques, we consider three variants of our approach: 1) without A2RL, 2) without DSIL, and 3) without EBC. Figure 3(b) shows the training curves. Each technique contributes to the overall performance. EBC and A2RL contribute more than DSIL. DSIL mainly boosts the performance for later sub-tasks, while A2RL and EBC have earlier effects on the overall pipeline. EBC contributes significantly, demonstrating that learning a robust policy is important for solving long-horizon tasks.

4.3 Visualization

To understand why our techniques work, we conduct an in-depth analysis. To understand the learned mask in A2RL, we compute saliency maps. For each action, we show the current state, the next state, and the saliency map of the learned mask on the current state. We find that the learned mask captures the dynamic information between two adjacent states, revealing curiosity on the effect of actions. The mask net learns to focus on uncertain parts of the current state. For the 'attack' action, the learned mask focuses on the objects in front of the agent. For the 'turn left' and 'turn down' actions, the mask net focuses on the parts that have major changes due to the rotation and translation of the agent's perspective. Our learned mask assists the agent in better understanding the 3D environment.

To understand how DSIL works, we visualize the state distribution that the agent visits. We compare PPO, PPO+SIL, and PPO+DSIL. At the early training stage, both methods explore randomly and sometimes reach the goal state successfully. After getting samples and training, PPO+DSIL starts to explore in a compact region, while PPO and PPO+SIL still explore in a wider region. DSIL pushes the agent to stay close to a good state distribution, reproducing its past behaviors and exploring in a better way, which incentivizes deep exploration for successful trajectories.

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

Baselines		2019 Competition Results	
Name	Score	Team Name	Score
SQIL	2.94	CDS (ForgER)	61.61
DQfD	2.39	mc$_r l$	42.41
Rainbow	0.42	I4DS	40.8
PDDDQN	0.11	CraftRL	23.81
BC	2.40	UEFDRL	17.9
		TD240	15.19
2020 Competition Results		2021 Competition Results	
Team Name	Score	Team Name	Score
HelloWorld (SEIHAI)	39.55	X3 (JueWu-MC)	76.97
michal_opanowicz	13.29	WinOrGoHome	22.97
NoActionWasted	12.79	MCAgent	18.98
Rabbits	5.16	sneakysquids	14.35
MajiManji	2.49	JBR_HSE	10.33
BeepBoop	1.97	zhongguodui	8.84

Table 2: The conditional success rate of each stage.

Methods	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6	Stage 7
SEIHAI	64%	78.6%	78.3%	84.7%	23%	0%	0%
JueWu-MC	92%	96%	96%	87%	46%	11%	0%

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 sophisticated tasks. Our novel techniques in representation learning and imitation learning improve both the performance and learning efficiency of the sub-policies. Experiments show that our pipeline outperforms all baseline algorithms and previous solutions from MineRL competitions. In future work, we would like to apply JueWu-MC to other Minecraft tasks, as well as other open-world games.