

An Empirical Evaluation of A Hierarchical Reinforcement Learning Method towards Modular Robot Control

Sho Takeda¹ · Satoshi Yamamori² · Satoshi Yagi¹ · Jun Morimoto^{1,2}

Received: date / Accepted: date

Abstract There is a growing expectation that deep reinforcement learning will enable multi-degree-of-freedom robots to acquire policies suitable for real-world applications. However, a robot system with a variety of components requires many learning trials for each different combination of robot modules. In this study, we propose a hierarchical policy design to segment tasks according to different robot components. The tasks of the multi-module robot are performed by skill sets trained on a component-by-component basis. In our learning approach, each module learns reusable skills, which are then integrated to control the whole robotic system. By adopting component-based learning and reusing previously acquired policies, we transform the action space from continuous to discrete. This transformation reduces the complexity of exploration across the entire robotic system. We validated our proposed method by applying it to a valve rotation task using a combination of a robotic arm and a robotic gripper. Evaluation based on physical simulations showed that hierarchical policy construction improved sample efficiency, achieving performance comparable to the baseline with 46.3% fewer samples.

Keywords Hierarchical Reinforcement Learning · Modular Reinforcement Learning · Multitask Manipulation · Modular Robot

1 Introduction

In actual applications, robotics tasks combine the various parts, requiring the robot to integrate its skills to

accomplish the complex task. Therefore, skill training of the modular robot, which has a conversion component, is anticipated. Specifically, to develop the learning framework of the modular robot, we focus on the arm and gripper combining robot, such that the robot can convert the gripper to the other to switch the skills of the entire robot. Although robot learning with deep reinforcement is energetically investigated, the sample efficiency problem still lies before us [1,2]. A robot system of multiple components addresses many samples to coordinate each part, leading to time-consuming training. For example, when components are swapped for different grippers, a robot must be retrained to consider the specific parts. As a result, considerable learning costs are consumed for each combination of robot components. However, a partial function of fewer components can perform the task in practical situations, and it is too redundant to retrain all combinations of components. In other words, each component has the potential to perform a specific task. We propose a hierarchical framework for each component to improve sample efficiency by learning policy.

Many studies have focused on methods that efficiently advance learning by introducing a hierarchical structure into the learning process. This approach is referred to as hierarchical reinforcement learning.

Conventional research on hierarchical reinforcement learning is classified based on tasks and agents into single-task, multi-task, and multi-agent scenarios. Firstly, consider the case where tasks and agents are single. Sutton et al. proposed introducing a hierarchical structure using the concept of Temporal Abstraction, which determines policies over multiple steps, defining Option [3] as actions at the upper level. There is an example of applying hierarchical reinforcement learning to robot control [4]. Some methods involve set-

¹ Graduate School of Informatics, Kyoto University, Kyoto, Japan

² Dept. of Brain Robot Interface, Computational Neuroscience Labs, ATR, Kyoto, Japan

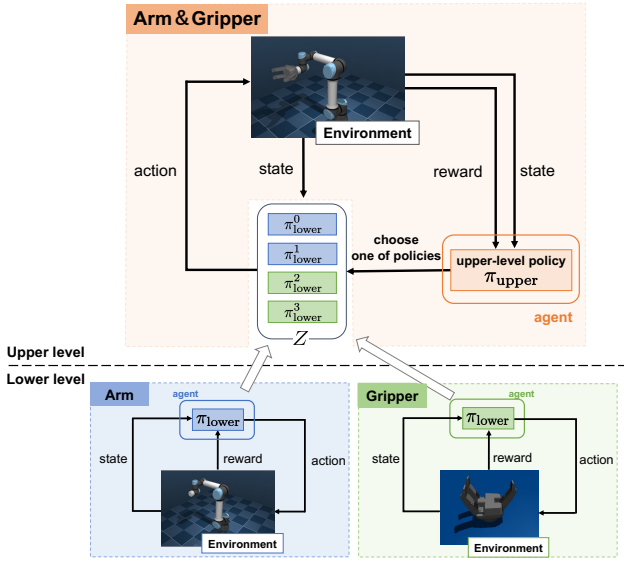


Fig. 1: Structure of the proposed hierarchical policy construction. Upper-level policy π_{upper} chooses the best policy based on the upper-level reward among the lower-level policies Z . Lower-level policy π_{lower}^i is trained to solve the task into which the whole robot task is segmented. Lower-level action/state space is shared with the upper-level action/state.

ting secondary goals when learning a task [5, 6], considering control of latent variables rather than directly contemplating actions [7–9]. Research is being conducted on multi-task hierarchical reinforcement learning, which involves learning reusable skills common to multiple tasks. Multiplicative Compositional Policies (MCP) [10] generate hierarchical policies by representing reusable skills using latent variables, allowing the generation of complex behaviors. Christen et al. propose Hierarchical Decomposition (HiDe) [11] for robot control, separating path planning from low-level control, and utilizing low-level control for multiple tasks. There is research in the field that combines hierarchical reinforcement learning with multi-agent reinforcement learning. Liu et al. have introduced a hierarchical structure to the policies of each agent by separating them into motion control and action planning [12]. By applying the machine learning technique Population Based Training (PBT) [13], they enable coordinated play in football matches by agents that have undergone learning. As mentioned above, hierarchical reinforcement learning has been applied to multitask and multi-agent. The robot, with combined components, incorporates the multi-agent property of individual components as agents with the multitask property of components function. Therefore, the cooperative learning

framework, by integrating the hierarchical structure, is inevitable for efficient learning.

In this study, we propose a hierarchical method for dividing tasks into individual components for robots (Fig. 1). The robot sequentially executes each component, like the valve manipulation task, which is reaching the valve with the arm before grasping the valve with the gripper. At that time, the only partial function of the specific component is activated in each segment of the task sequence. Therefore, the robot with combined components can accomplish the sequential task by training the sub-task component-wise at a lower level and optimizing the appropriate combination of the lower-level policies. This method can be formulated using two-stage reinforcement learning, obtaining the lower-level policy for each component and the upper-level policy for suitable combinations throughout the deep learning framework. We demonstrated that the proposed method utilizes the relationship between robot components and segmented tasks. Compared to the non-hierarchical approach, the hierarchical policy construction enabled the acquisition of a policy that effectively grasped the valve. Although the valve was not effectively grasped in the non-hierarchical approach, the hierarchical policy construction enabled the acquisition of a policy that grasped the valve effectively and improved sample efficiency compared with the non-hierarchical approach.

2 Methods

We propose a hierarchical skill learning framework for dividing the task of the whole module into multiple tasks for each robot component, as shown in Fig. 1. The skills learned for each component is reusable for the robot with a combined arm and gripper, such as the reaching task in the arm or the grasping task in the gripper component. Therefore, the learning phase is divided into two stages: the lower level and the upper level.

Initially, each component learns the reusable skill—the lower policy π_{lower} . For example, the gripper learns to grasp objects and rotate handles. Lower-level policies are learned by moving one component while keeping others fixed. After training the lower-level policies, the upper policy π_{upper} is learned to achieve each task by using these trained lower-level policies. The upper-level policy selects the best skill from the set of lower-level policies Z . Expanding this set enhances the robot’s overall skill set, enabling it to execute a wider range of tasks.

Our method solves the two-stage optimization problem, which extends the continuous reinforcement learn-

ing algorithm, Soft Actor-Critic (SAC [14, 15]), described below:

$$\max_{\pi_{\text{upper}}} E_{\tau} \left[\sum_{t \geq 0} \gamma^t (r_{\text{upper}}(s, a_t) - \alpha \log \pi_{\text{upper}}) \right], \quad (1)$$

$$\max_{\pi_{\text{lower}}} E_{\tau} \left[\sum_{t \geq 0} \gamma^t (r_{\text{lower}}(s, a_t) - \alpha \log \pi_{\text{lower}}) \right], \quad (2)$$

$$a_t \sim \pi_{\text{lower}}^i(A|s), w \sim \pi_{\text{upper}}(W|s),$$

$$i = \arg \max_{k \in \{0, 1, \dots, |Z|\}} w_k,$$

where $\tau = \{\dots, s(t), a_t(t), i(t), \dots\}_{t \geq 0}$ is the trajectory for each time step t ; the $\pi_{\text{upper}}/\pi_{\text{lower}}$ are upper/lower policy; $s \in S$, $a_t \in A$, and $i \in \{0, 1, \dots, |Z|\}$ are state, the reusable action (lower action) and upper action; the number of lower policies is M ; $r_{\text{upper}}/r_{\text{lower}}$ is the upper/lower reward; policy weight $w \in W$ is the output of upper-level policy and the elements of w correspond to the index z of each lower-level policy; and α and γ are the hyperparameters of the SAC. The neural network embodies the upper/lower-level policies and is trained by the SAC algorithm [15] from collected data through the interaction for an environment.

The lower level cuts out the reward r_{lower} from the upper level for each component and independently learns the policy for reusable actions. For example, environments are constructed separately to control the robot arm and control only the robot gripper, allowing agents to learn each task independently. Consequently, this learning process provides policies to achieve reach tasks with the robot arm and grasp tasks with the gripper.

The upper-level policy π_{upper} at each timestep selects the best skill from lower-level policies Z , which obtains the maximum upper reward r_{upper} in Eq. (1). The lower action a is reused for the direct action to the robot because the whole robot combines with lower components.

3 Experiments

This section details the experimental setup and task conditions used to evaluate the proposed two-stage learning method. The upper and lower levels each learn separate tasks, described below.

3.1 Environmental Settings

We demonstrated the hierarchical policy construction by the valve manipulation task in MuJoCo simulations

[16]. The whole robot system consisted of the 6DoF Arm robot from Universal Robots [17] with the 3-Finger Adaptive Robot Gripper from Robotiq [18] as shown in Fig. 2. We divided the robot at the wrist into arm and the gripper components since the valve manipulation task is constructed from the reaching and grasping tasks. The arm has six joints with PD position servos, and the gripper can control the scale of open/closed from 0 to 1. The 6th joint is included in the gripper component because this axis must be used to rotate a valve.

Details of observation and action variables are shown in Table 1 and 2.

Table 1: Observation Variable

Component	name	range
Arm	Joint $\theta_1 \sim \theta_5$	$(-\pi, \pi)^5$
	Joint velocity $\omega_1 \sim \omega_5$	$(-\pi, \pi)^5$
	Gripper position $\mathbf{p}_{\text{gripper}}$	$\mathbf{R}^3$
Gripper	Joint θ_{gripper}	$(-2\pi, 2\pi)$
	Joint velocity ω_{gripper}	$(-2\pi, 2\pi)$
	open/closed c	$[0, 1]$
	Valve angle ϕ	$(-2\pi, 2\pi)$
Upper level	Gripper position $\mathbf{p}_{\text{gripper}}$	$\mathbf{R}^3$
	Valve angle ϕ	$(-2\pi, 2\pi)$

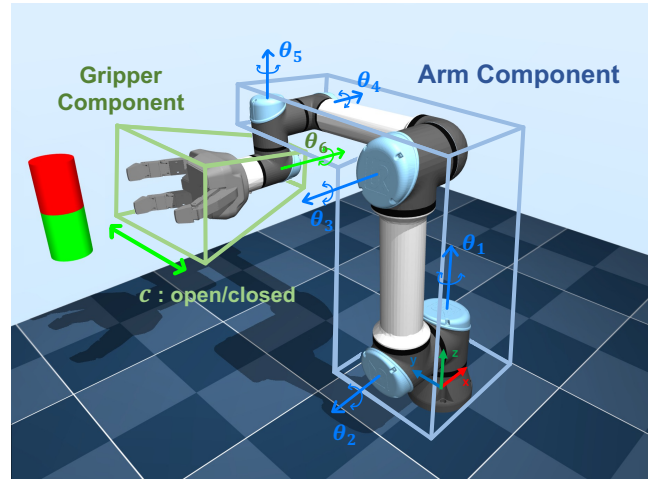


Fig. 2: Robot with combined an arm and a gripper component to manipulate valve. The arm component comprises five joints of UR5e. The gripper component comprises one wrist joint of UR5e and a 3-finger Adaptive Robot Gripper. A two-tone coloring cylinder is a valve.

Table 2: Action Variable

Component	name	range
Arm	Joint $a_1 \sim a_5$	$(-\pi, \pi)^5$
Gripper	Joint a_6	$(-2\pi, 2\pi)$
	open/closed a_7	$[0, 1]$

3.2 Upper-level Task

The valve manipulation task demands that the robot bring the end-effector to the valve position and rotate the valve to match a target angle. We conducted four tasks, each with two settings for valve position and target angle. The target valve positions $\mathbf{p}_{\text{valve}}$ are set from the base of the arm as $(-0.5, -0.4, 0.5)$ or $(-0.8, 0.2, 0.5)$ [m]. The target angles ϕ_{ref} were set to π or $-\pi$. The reward function was designed considering the distance from the valve angle and the difference from the target angle, as follows:

$$r^{\text{upper}} = -\text{HuberLoss}(\mathbf{p}_{\text{gripper}} - \mathbf{p}_{\text{valve}}) - 0.1\text{HuberLoss}(\phi_{\text{ref}} - \phi), \quad (3)$$

where the gripper position is $\mathbf{p}_{\text{gripper}}$ and the valve angle is ϕ . The policy neural network is a multilayer perceptron (MLP) with two depths and 32 widths. Its output is the four-dimensional array of the selection weight used to choose the lower-level policy. At each timestep t , the upper policy selects the specific lower policy corresponding to the maximum index in the weight array. Not selected policies are given state input at that previous timestep to maintain the action. The learning data is repeatedly collected from the episode trajectory for 20 seconds with a 400 ms control period.

3.3 Low-level Task

The reaching and grasping tasks are lower-level tasks segmented from the valve manipulation task. The hierarchical policy construction separately solves these tasks to obtain the reusable skill.

3.3.1 Reaching Task

The reaching task demands that an arm brings the end-effector to a valve position with a suitable orientation. Two targets were designated valve positions starting from the same initial position, as explained in Section 3.2. Allowing sufficient time for task completion, episodes were terminated after the 20s. The gripper component actions were given input to maintain the

initial state. The reward function was designed as follows:

$$r^{\text{reach}} = -\text{HuberLoss}(\mathbf{p}_{\text{gripper}} - \mathbf{p}_{\text{valve}}) - 2\text{HuberLoss}(\mathbf{t}_{\text{gripper}} - \mathbf{h}) - 5 \cdot 10^{-4} \|\boldsymbol{\omega}_{\text{arm}}\|^2 - 0.05 \|\mathbf{a}(n) - \boldsymbol{\theta}(n+1)\|^2, \quad (4)$$

where the valve position is $\mathbf{p}_{\text{valve}} = (-0.5, -0.4, 0.5)$ or $(-0.8, 0.2, 0.5)$, the action in other words target joint angle is $\mathbf{a}_{\text{arm}}$, the joint angle and angular velocity are $\boldsymbol{\theta}_{\text{arm}}$ and $\boldsymbol{\omega}_{\text{arm}}$, the gripper and target orientation (Euler angles) are $\mathbf{h}$ and $\mathbf{t}_{\text{gripper}} = (\pi/2, 0, 0)$ or $(0, 0, 0)$. The last term is added to suppress the arm vibration by reducing the difference between action $\mathbf{a}(n)$ and one-step future joint angle $\boldsymbol{\theta}(n+1)$. The policy neural network is an MLP with two depths and 256 widths and outputs the target angle for each joint.

3.3.2 Grasping Task

The grasping task demands that the valve angle approaches the target angle. The initial end-effector position was set with the valve position to train only the gripper open/closed and valve rotation. The target angles ϕ_{ref} were trained separately as π or $-\pi$. Like the reaching case, the arm component actions were given input to maintain the initial state. We designed the reward function described as follows:

$$r^{\text{grasp}} = -\text{HuberLoss}(\phi_{\text{ref}} - \phi) + 0.1c, \quad (5)$$

where the target valve angle is ϕ_{ref} and the open/closed factor is c . The last term is to induce the gripper closing, i.e., $c = 1$. The policy structure is the same MLP as the reaching task.

4 Results

This section validates the skill obtained for the valve manipulation task from the hierarchical structure and sample efficiency viewpoint. Fig. 3a shows the snapshot while the upper policy chooses the reusable action for the valve manipulation task. The selected policy was switched from reaching one policy to grasping one policy at about 3.5 seconds, as shown in Fig. 4. The upper policies have selected the correct one among the four lower policies.

We also evaluated the non-hierarchical policy as a baseline to confirm the superiority of the proposed method. The non-hierarchical policy was trained by the SAC algorithm given the state of the arm and gripper components and outputs all actions of the whole robot.

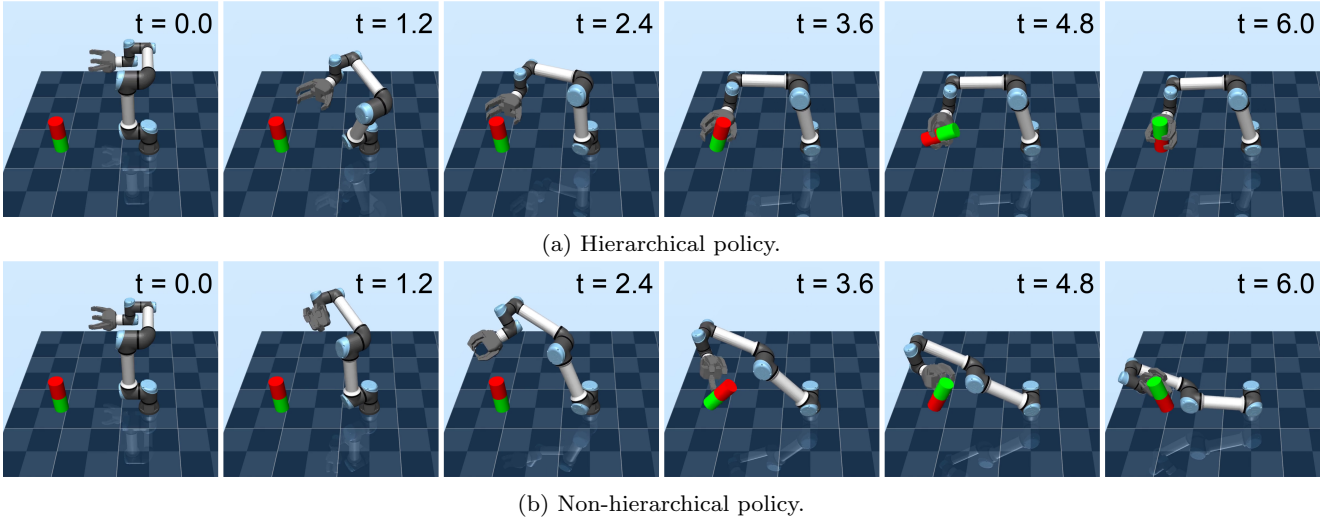


Fig. 3: Snapshot sequences of the valve manipulation experiment in the upper-level learning. The hierarchical policy switched the reusable action from reaching the valve until $t = 3$ to rotating the valve until $t = 6$. The non-hierarchical policy couldn't grasp the valve effectively.

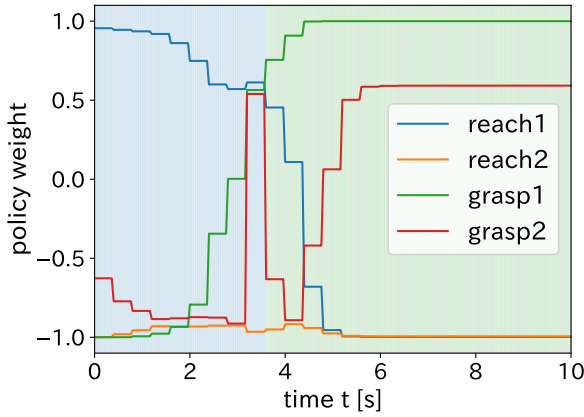


Fig. 4: Transition of action of upper-level policy. The upper-level policy outputs weight as action and selects the largest weighted one from four lower policies. Background color indicates the selected policy for each time step.

The reward function utilized r^{upper} from Eq. (3) used in the upper-level learning. A comparison of the learning curves is illustrated in Fig. 5 for different combinations of the target valve position and angle. The hierarchical policy construction rapidly outperforms the returns from non-hierarchical. Policies obtained from learning without a hierarchical structure did not yield strategies to grasp the valve in any of the tasks.

We compared the number of samples required to achieve the maximum return in learning policies with-

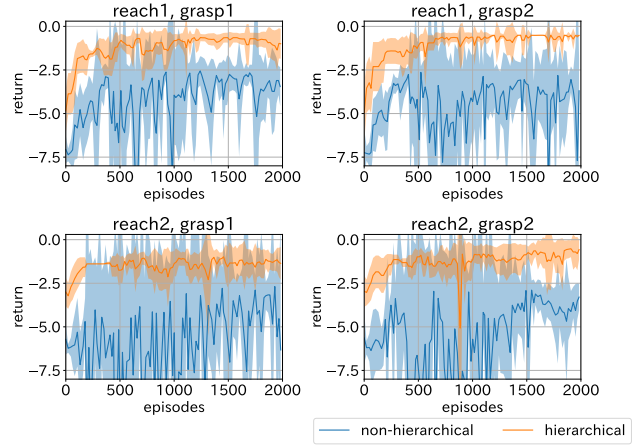


Fig. 5: Learning curves comparing performance between hierarchical and non-hierarchical. This solid line shows the average of 10 times the learning process. Each subplot shows a total of four learning curves with two target positions $\mathbf{p}_{\text{valve}}$ and two target angles ϕ_{ref} . Evaluations are conducted every 20 episodes.

out hierarchy. Since the non-hierarchical policy could not achieve the task, we compared the number of samples required to achieve the maximum return rather than the number needed to accomplish the task. This comparison is illustrated in Fig. 6. Upon conducting ten learning iterations for each task and comparing hierarchical learning with non-hierarchical learning, we obtained policies with equal returns at an average of 46.3% fewer samples. These results suggested that the

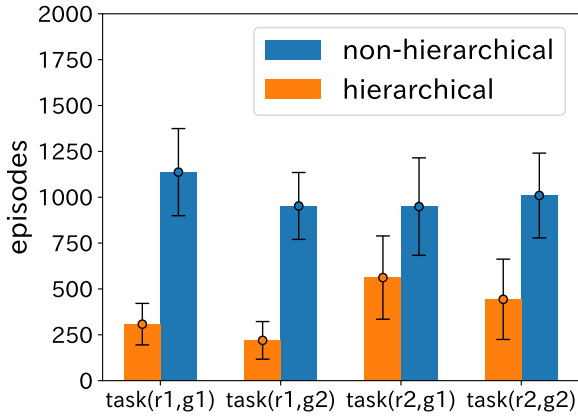


Fig. 6: Number of episodes required to achieve maximum hierarchical and non-hierarchical learning return. The error bar shows the standard error of the number of samples.

proposed method facilitates the learning of manipulation tasks. In the lower-level learning phase, policies are acquired by setting tasks individually for each component. The improvement in learning performance of the hierarchical approach is thought to be due to the fact that the exploration by the upper-level learning system has decreased as a result of component-wise learning at the lower level.

5 Conclusions

We proposed a hierarchical approach that segments tasks according to different robot components and evaluated our method on a robot composed of an arm and a gripper. At the lower level, it learns to reach tasks for the robot arm and grasp tasks for the robot gripper. At the upper level, it learns multiple tasks of grasping valves and rotating them, setting different valve positions and target angles for each. The results of the upper-level learning demonstrated that the upper-level policy can appropriately select policies from the lower-level policies. Additionally, compared to learning policies without a hierarchical structure, it confirms that constructing hierarchical policies promotes the learning of manipulation actions. In this research, the necessity of increasing the number of low-level policies when adding tasks represents a limitation. Integrating this approach with continual learning [19, 20] would likely lead to more efficient learning process.

Acknowledgements This work was supported by JST Moonshot R&D program JPMJMS223B-3, by project

JPNP20006 commissioned by NEDO, and by JSPS KAKENHI 22H03669.

References

1. Brohan A, Brown N, Carbajal J, Chebotar Y, Dabis J, et al (2023) RT-1: Robotics transformer for real-world control at scale. *Proceedings of Robotics: Science and Systems*
2. Gu S, Holly E, Lillicrap T, Levine S (2017) Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. *2017 IEEE International Conference on Robotics and Automation (ICRA)* pp. 3389–3396
3. Sutton RS, Precup D, Singh S (1999) Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence* 112(1-2):181–211
4. Morimoto J, Doya K (2001) Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning. *Robotics and Autonomous Systems* 36(1):37–51
5. Frans K, Ho J, Chen X, Abbeel P, Schulman J (2018) Meta learning shared hierarchies. *International Conference on Learning Representations*
6. Nachum O, Gu S, Lee H, Levine S (2018) Data-Efficient Hierarchical Reinforcement Learning. *Advances in Neural Information Processing Systems*
7. Osa T, Tangkaratt V, Sugiyama M (2018) Hierarchical Reinforcement Learning via Advantage-Weighted Information Maximization. *International Conference on Learning Representations*
8. Haarnoja T, Hartikainen K, Abbeel P, Levine S (2018) Latent space policies for hierarchical reinforcement learning. *International Conference on Machine Learning* pp. 1851–1860
9. Peng XB, Guo Y, Halper L, Levine S, Fidler S (2022) ASE: Large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Transactions On Graphics (TOG)* 41(4):1–17
10. Peng XB, Chang M, Zhang G, Abbeel P, Levine S (2019) MCP: Learning Composable Hierarchical Control with Multiplicative Compositional Policies. *Advances in Neural Information Processing Systems* 32
11. Christen S, Jendele L, Aksan E, Hilliges O (2021) Learning functionally decomposed hierarchies for continuous control tasks with path planning. *IEEE Robotics and Automation Letters* 6(2):3623–3630
12. Liu S, Lever G, Wang Z, Merel J, Eslami SMA, et al (2022) From motor control to team play in simulated humanoid football. *Science Robotics* 7(69)
13. Jaderberg M, Dalibard V, Osindero S, Czarnecki WM, Donahue J, et al (2017) Population based training of neural networks. *arXiv preprint arXiv:1711.09846*
14. Haarnoja T, Zhou A, Abbeel P, Levine S (2018) Soft actor-critic: Off-Policy Maximum entropy deep reinforcement learning with a stochastic actor. *International Conference on Machine Learning* pp. 1861–1870
15. Haarnoja T, Zhou A, Hartikainen K, Tucker G, Ha S, et al (2018) Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*
16. Todorov E, Erez T, Tassa Y (2012) Mujoco: A physics engine for model-based control. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* pp. 5026–5033

17. Zakka K, Tassa Y, MuJoCo Menagerie Contributors (2023) MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo. https://github.com/google-deepmind/mujoco_menagerie. Accessed Dec 20 2023
18. Franceschetti A (2023) Universal Robots UR5 + Robotiq S Model 3 Finger Gripper. <https://roboti.us/forum/index.php?resources/universal-robots-ur5-robotiq-s-model-3-finger-gripper.22/>. Accessed Dec 20 2023
19. Lesort T, Lomonaco V, Stoian A, Maltoni D, Filliat D, et al (2020) Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion* 58:52–68
20. Auddy S, Hollenstein J, Saveriano M, Rodríguez-Sánchez A, Piater J (2023) Continual learning from demonstration of robotics skills. *Robotics and Autonomous Systems* 165:104427