ARC-RL: SELF-EVOLUTION CONTINUAL REINFORCEMENT LEARNING VIA ACTION REPRESENTATION SPACE

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual Reinforcement Learning (CRL) is a powerful tool that enables agents to learn a sequence of tasks, accumulating knowledge learned in the past and using it for problem-solving or future task learning. However, existing CRL methods all assume that the agent's capabilities remain static within dynamic environments, which doesn't reflect real-world scenarios where capabilities evolve. This paper introduces *Self-Evolution Continual Reinforcement Learning* (SE-CRL), a new and realistic problem where the agent's action space continually changes. It presents a significant challenge for RL agents: How can policy generalization across different action spaces be achieved? Inspired by the cortical functions that lead to consistent human behavior, we propose an **A**ction **R**epresentation **C**ontinual **R**einforcement **L**earning framework (ARC-RL) to address this challenge. Our framework builds an action representation space by self-supervised learning on transitions, decoupling the agent's policy from the specific action space. For a new action space, the decoder of the action representation is expanded or masked for adaptation and regularized fine-tuned to improve the stability of the policy. Furthermore, we release a benchmark based on MiniGrid and Procgen to validate the effectiveness of methods for SE-CRL. Experimental results demonstrate that our framework significantly outperforms popular CRL methods by generalizing the policy across different action spaces. [1]

1 INTRODUCTION

Continual Reinforcement Learning (CRL, a.k.a. lifelong reinforcement learning) is an emerging research field that aims to emulate the human capacity for lifelong learning and tackles the challenges of long-term, real-world applications characterized by diversity and non-stationarity (Rolnick et al., 2019; Kessler et al., 2022). Specifically, CRL extends traditional Deep Reinforcement Learning (DRL) by empowering agents with the ability to learn from a sequence of tasks, preserving knowledge from previous tasks, and using this knowledge to enhance learning efficiency and performance on future tasks. Although CRL research requires the agent's ability to adapt to dynamic environments (Khetarpal et al., 2022), it typically assumes that the agent's capabilities (action space) remain static while the external environment changes. This assumption does not reflect realistic situations where an agent's capabilities may evolve. Living systems not only need to adapt to radical changes in the environment (Emmons-Bell et al., 2019), but also need to deal with changes to their structure and function (Blackiston et al., 2015). Similarly, continual learning agents also need to deal with their evolving capabilities (Kudithipudi et al., 2022). For example, the action space of agents

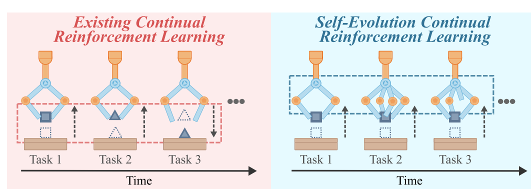


Figure 1: An example of two problems. **Existing CRL**: A robot uses two fingers to grasp objects while the objects or grasping way changes. **SE-CRL**: A robot initially trained with two fingers is upgraded to four fingers or loses a finger but must continue grasping objects.

[1]Code are released in Supplementary Material.

in real-world applications may change due to software or hardware updates (Wang et al., 2019; Ding et al., 2023) or damages (Kriegman et al., 2019; Kwiatkowski & Lipson, 2019). Therefore, continual learning with the changes of action space is crucial for developing more sophisticated and adaptable artificial intelligence systems.

While existing research in RL (Chandak et al., 2020; Ding et al., 2023) has made initial explorations into the challenges posed by changing action spaces, these studies have certain limitations. Specifically, they primarily focus on continual adaptation without addressing other critical issues in CRL, such as catastrophic forgetting. Additionally, they only consider expanding action spaces, neglecting other types of changes in the action space. Building on these foundational studies, we propose a new and more general problem called *Self-Evolution Continual Reinforcement Learning* (SE-CRL), where the agent needs to continual learning with its evolving capabilities. Figure 1 illustrates the difference between SE-CRL and existing CRL. While existing CRL requires exploring how to respond to dynamic environmental changes, SE-CRL needs to maintain the agent's performance as the capabilities evolve, considering catastrophic forgetting and knowledge transfer. SE-CRL supplements existing CRL research by considering dynamics in a broader context. As an early step, this work focuses on discrete action spaces and assumes that the task logic remains unchanged over time.

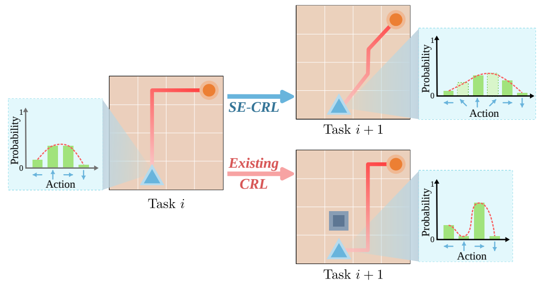


Figure 2: Different challenges of two problems. **Existing CRL**: After the environment changes, the number of actions remains constant (number of columns), while the probability distribution shifts significantly (trend of the red line). **SE-CRL**: After the action space changes, the number of actions changes, while the probability distribution is relatively stable.

As shown in Figure 2, the main challenge of SE-CRL is different from existing CRL. The main challenge of existing CRL is dealing with the significant shift of the probability distribution of the actions after the environment changes, while the main challenge of SE-CRL is to cope with changes in the actions' number after the action space changes. Although a general policy can be obtained using the union of all action spaces, the previous global optimum may become a local one that does not fit the new action space. This process, however, underscores the crucial role of expertise, as it requires prior knowledge about all action spaces. In summary, SE-CRL can be formally modeled as the following problem: *How to achieve policy generalization across different action spaces with the same task logic?*

Animals, including humans, consistently perform behaviors even years after learning (Georgopoulos & Pellizzer, 1995; Emmons-Bell et al., 2019; Blackiston et al., 2015). It is due to the brain's ability to represent actions in a latent space, allowing for the generalization across different contexts. Precisely, the stability of latent dynamics of neural activity reflects a fundamental feature of learned cortical function, leading to stable and consistent behavior (Gallego et al., 2020). In addition, the research on self-supervised learning for reinforcement learning has been shown to be effective in improving the generalization ability of the agent (Chandak et al., 2019; Liu et al., 2024; Fang & Stachenfeld, 2024).

Inspired by these, we propose an **A**ction **R**epresentation **C**ontinual **R**einforcement **L**earning framework (ARC-RL) to address the challenge of SE-CRL by generalizing policy across action spaces with different sizes. ARC-RL first learns an action representation space by learning a pair of encoder and decoder. They are trained through self-supervised learning on transitions collected from the agent's exploration of the environment. The encoder maps the agent's actions to action representations, and the decoder maps them to action probabilities. Once trained, the encoder and the decoder are fixed, and the agent's policy is trained based on the action representation space. When the action space changes, the decoder's structure is updated to accommodate the size of the new action space. The agent then explores the environment with the new action space and fine-tunes the encoder and the decoder, adding regularization for the fine-tuning of the decoder to maintain stability. In this process, the function of the decoder is similar to that of the cerebellum of humans, while the policy corresponds to that of the primary motor cortex. The former is essential for learning new mapping, but the latter is vital for consolidating the new mapping (long-term retention) (Haar et al., 2015; Gazzaniga et al., 2019; Weightman et al., 2023).

To evaluate the performance of CRL methods in SE-CRL, we release a benchmark based on MiniGrid (Chevalier-Boisvert et al., 2023) and Procgen (Cobbe et al., 2020), which includes two sets of tasks with different action spaces and three task sequence situations (Expansion, Contraction, and their combinations) designed to test the agent's generalization ability. Experimental results demonstrate that ARC-RL effectively handles SE-CRL compared to popular CRL methods.

Our contributions can be summarized as follows:

- To the best of our knowledge, we are the first to formally propose the *self-evolution continual reinforcement learning* problem (SE-CRL), supplementing the existing CRL by focusing on the agent's evolving capabilities.
- We propose a CRL framework called ARC-RL, which builds an action representation space and uses regularized fine-tuning to address the challenges of SE-CRL.
- We release a benchmark of SE-CRL to evaluate the performance of CRL methods. Experiments show that ARC-RL is more effective compared to the others.

2 Related Works

2.1 Self-Supervised Learning for Reinforcement Learning

Existing reinforcement learning methods often require extensive data interactions with the environment, particularly in image-based RL tasks, which suffer from low sample efficiency and generalizability (Schrittwieser et al., 2020; Ye et al., 2020; Wang et al., 2024b). Recently, Self-Supervised Learning (SSL) has emerged to address these issues by learning a compact and informative representation of the environment (Li et al., 2022; Stooke et al., 2021). SSL approaches in RL encompass auxiliary tasks, contrastive learning, and data augmentation, each contributing to improved performance and efficiency.

Auxiliary tasks in SSL for RL involve learning additional objectives that aid in representation learning. These tasks include reconstruction loss Chandak et al. (2019); Liu et al. (2024), world modeling (Hafner et al., 2020), and information-theoretic techniques (Pong et al., 2020) to obtain efficient representations. Contrastive learning has gained traction in RL for its ability to learn valuable representations without requiring labeled data (Laskin et al., 2020a). Additionally, contrastive learning has shown success in goal-conditioned RL tasks without needing extra data augmentation or auxiliary objectives (Eysenbach et al., 2022). Data augmentation strategies, as demonstrated by DrQ, apply simple image augmentations to standard model-free RL algorithms, enhancing robustness and efficiency (Yarats et al., 2021) RAD further explores data augmentations for both pixel-based and state-based inputs, significantly improving data efficiency and generalization (Laskin et al., 2020b).

While SSL for RL has significantly improved sample efficiency and generalization, the open research challenge of using SSL in CRL is an intriguing area that requires further exploration. Our proposed framework uses self-supervised learning to build an action representation space that decouples the agent's policy from the specific action space, enabling policy generalization.

2.2 Continual Reinforcement Learning

Continual reinforcement learning focuses on training RL agents to learn multiple tasks sequentially without prior knowledge, generating significant interest due to its relevance to real-world artificial intelligence applications (Khetarpal et al., 2022).

A central issue in CRL is catastrophic forgetting, which has led to various strategies for knowledge retention. PackNet and related pruning methods (Mallya & Lazebnik, 2018; Schwarz et al., 2021) preserve model parameters but often require knowledge of task count. Experience replay techniques such as CLEAR (Rolnick et al., 2019) use buffers to retain past experiences but face memory scalability challenges. In addition, some methods prevent forgetting by maintaining multiple policies or a subspace of policies (Schöpf et al., 2022; Gaya et al., 2022). Furthermore, task-agnostic CRL research indicates that rapid adaptation can also help prevent forgetting (Caccia et al., 2023).

Another issue in CRL is transfer learning, which is crucial for efficient policy adaptation. Naive approaches, like fine-tuning, train a single model on each new task and provide good scalability and transferability but suffer from catastrophic forgetting. Regularization-based methods, such as EWC (Kirkpatrick et al., 2017; Wang et al., 2024a), have been proposed to prevent this side effect, but often reduce plasticity. Some architectural innovations have been proposed to balance the trade-off between plasticity and stability (Rusu et al., 2016; Berseth et al., 2022). Furthermore, methods like OWL (Kessler et al., 2022) and MAXQINIT (Abel et al., 2018) leverage policy factorization and value function transfer, respectively, for improved learning across tasks.

Most existing methods perform well when applied to sequences of tasks with static agent capabilities and dynamic environments, such as when environmental parameters are altered, or the objectives within the same environment are different (Pan et al., 2024). However, their effectiveness is greatly diminished when the agent's capabilities evolve. Our proposed framework aims to overcome this limitation by building an action representation space.

Additionally, LAICA (Chandak et al., 2020) and DAE (Ding et al., 2023) are particularly relevant in this context. LAICA primarily addresses changes in the action space but focuses on expansion rather than contraction and others, and does not account for catastrophic forgetting. DAE investigates incremental reinforcement learning with expanding action spaces and state spaces but also lacks consideration of more complex action space changes and the critical aspects of CRL. In contrast, our work proposed a more general problem in the context of CRL, considering both the expansion and contraction situations of the action space and the catastrophic forgetting problem.

3 Self-Evaluation Continual Reinforcement Learning

3.1 Preliminaries

The reinforcement learning process can be formulated as a Markov Decision Process (MDP) $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}\}$. A MDP represents a problem instance that an agent needs to solve over its lifetime. Here, $\mathcal{S}$ and $\mathcal{A}$ denote the state and action space, respectively, while $\mathcal{P} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the transition probability function, and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to [r^{\min}, r^{\max}]$ is the reward function. At each time step, the learning agent perceives the current state $S_t \in \mathcal{S}$ and selects an action $A_t \in \mathcal{A}$ according to its policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$. The agent then transitions to the next state $S_{t+1} \sim \mathcal{P}(\cdot|S_t, A_t)$ and receives a reward $R_t = \mathcal{R}(S_t, A_t, S_{t+1})$.

The state-action value function for policy π is $Q^\pi(s, a) = \mathbb{E}_\pi\left[\sum_{j=0}^{H-t} \gamma^j R_{t+j}|S_t = s, A_t = a\right]$, where γ is the discount factor of the reward, and H is the horizon. Following the Bellman equation $V^\pi(s) = \sum_{a\in\mathcal{A}} \pi(s|a)Q^\pi(s, a)$, the state value function can be formulated as $V^\pi(s) = \mathbb{E}_\pi\left[\sum_{j=0}^{H-t} \gamma^j R_{t+j}|S_t = s\right]$. The goal of an agent is to find an optimal policy π^* to maximize the expected return $\mathbb{E}_{\pi^*}\left[\sum_{t=0}^{H} \gamma^t \mathcal{R}(S_t, A_t, S_{t+1})\right]$, which is the value function of the initial state.

3.2 Problem Formalization

In real-world scenarios, the capabilities of an agent may evolve over time. To explore this, we introduce a new problem called *Self-Evolution Continual Reinforcement Learning (SE-CRL)*. This problem can be formally defined as a sequence of Markov Decision Processes (MDPs) $\{(\mathcal{S}, \mathcal{A}^i, \mathcal{P}^i, \mathcal{R})|i = 1, 2, ..., N\}$, where N is the total number of MDPs and $\mathcal{A}^i$ represents the action space available to the agent at MDP i. Following the convention in CL, we still use "task" to represent each MDP in the sequence. Each task in the sequence shares a common state space $\mathcal{S}$ and reward function $\mathcal{R}$, but differs in the action space and implicitly in the transition probability function $\mathcal{P}^i$, which is influenced by the action space $\mathcal{A}^i$. To simplify the problem, we assume that the action space is discrete and finite, and we focus on the impact of changing action spaces on the learning process while assuming $\mathcal{P}^i$ remains conceptually similar across tasks.

Then, the dynamics of the action space can be characterized by differences in successive action spaces. For each task $i > 1$, the action space $\mathcal{A}^i$ can be related to the previous action space $\mathcal{A}^{i-1}$ in one of the following situations ($\mathcal{A}^i \neq \mathcal{A}^{i-1}$):

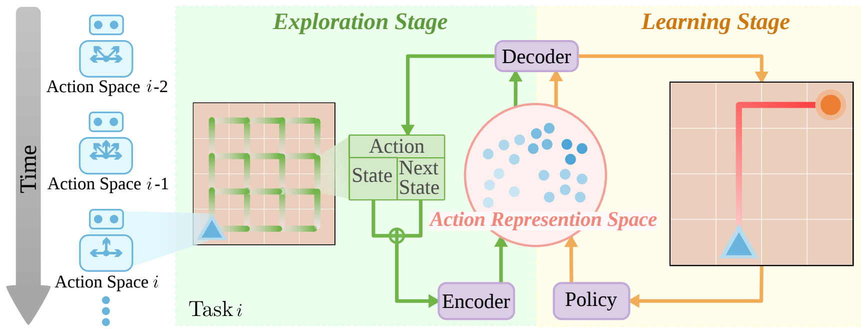


Figure 3: The overview of the proposed framework. The tasks with different action spaces are learned sequentially. Each task consists of two stages: the exploration stage (**green**) and the learning stage (**yellow**). The former aims to build an action representation space, and the latter aims to learn a policy based on the learned space.

1. **Expansion**: $\mathcal{A}^{i-1} \subset \mathcal{A}^i$ (new actions are added).
2. **Contraction**: $\mathcal{A}^{i-1} \supset \mathcal{A}^i$ (some actions are removed).
3. **Partial Change**: $\mathcal{A}^{i-1} \cap \mathcal{A}^i \neq \emptyset$ and $\mathcal{A}^{i-1} \not\subseteq \mathcal{A}^i$ and $\mathcal{A}^i \not\subseteq \mathcal{A}^{i-1}$ (some actions are removed and some actions are added).
4. **Complete Change**: $\mathcal{A}^{i-1} \cap \mathcal{A}^i = \emptyset$ (all actions are removed and new actions are added).

Previous work focuses on the first situation from the perspective of transfer reinforcement learning (Chandak et al., 2020; Ding et al., 2023). However, the other situations are less explored in the literature, especially in the broader context of CRL. In this work, we take a step further to address the problem of SE-CRL by considering the first two situations and their combinations.

The policy of the RL agent on task i is denoted as $\pi_{\theta^i} : \mathcal{S} \times \mathcal{A}^i \rightarrow [0, 1]$, where θ^i represents the policy parameters. After learning on tasks $\{1, 2, \cdots, i\}$, the agent's objective is to learn a policy that maximizes the average expected return overall tasks. This can be formally expressed as:

$$\max_{\theta^i} \frac{1}{i} \sum_{j=1}^{i} \mathbb{E}_{\pi_{\theta^i}} \left[\sum_{t=0}^{H^j} \gamma^t \mathcal{R}(S_t, A_t^j, S_{t+1}) \right], \tag{1}$$

where H^j is the horizon of task j, and $A_t^j \in \mathcal{A}^j$ is the action at the t-th step on task j. The expected return at each task is related to the current policy π_{θ^i} and the corresponding action space $\mathcal{A}^j$.

4 Action Representation Continual Reinforcement Learning

4.1 Framework

Our goal is to design a framework for generalized policy learning that can adapt to the changing action space, enhancing agent adaptation to evolving action capabilities. Recent neuroscience research has shown that the stability of latent dynamics of neural activity reflects a fundamental feature of learned cortical function that leads to long-term and consistent human behavior (Gallego et al., 2020). Furthermore, research on SSL for RL has demonstrated its effectiveness in improving the agent's generalization ability (Chandak et al., 2019; Liu et al., 2024; Fang & Stachenfeld, 2024). Drawing inspiration from these findings, we propose a new framework, named **A**ction **R**epresentation **C**ontinual **R**einforcement **L**earning (ARC-RL), to enables the agent to generalize policy across different action spaces.

Figure 3 illustrates the overview of ARC-RL. By decoupling the policy of the agent from the action space, the policy can be generalized to new action spaces efficiently. The interaction between the agent and the environment at each task is achieved through the action representation space. Each task in ARC-RL consists of a two-stage process: **Exploration Stage)** As shown in the left part of Figure 3, the agent explores the current action space in the environment, collects the transitions (state-action-state pairs), and learns an encoder-decoder pair through SSL. The encoder maps the action space to an action representation space, and the decoder maps the action representation space to the action space. When the action space changes, the agent can adapt by modifying the decoder's structure, while the parameters of the encoder and the decoder are updated. Furthermore, to maintain the stability of the policy, the regularization term is added to the fine-tuning process of the decoder. **Learning Stage)** As shown in the right part of Figure 3, the agent learns a policy based on the learned action representation space, rather than the specific action space of each task. Specifically, the action representation space is treated as the action space, and a standard RL policy is used to maximize the expected return. Once the action space changes, the agent merely needs to use the updated decoder to interact with the environment. In this way, the policy can maintain stability through the action representation space, and the agent can adapt to the new action space efficiently.

4.2 ACTION REPRESENTATION SPACE BUILDING

We use self-supervised learning to build an action representation space for the agent. The agent explores the action space in the environment of task i before learning the policy. It collects the transitions $\mathcal{T}^i = \{(s, a, s')_m | m = 1, 2, \cdots, M\}$ without reward, where s' is the next state of s after taking action a and M is the number of transitions. Based on the work of auxiliary tasks in reinforcement learning (Chandak et al., 2019; Fang & Stachenfeld, 2024), we believe that the features of the actions can be naturally represented by their influences of state changes. Therefore, the auxiliary task of the action representation is to predict the next state s' given the current state s and the action a. Specifically, for a transition (s, a, s'), the encoder f_{ϕ^i} parameterized by ϕ^i maps an action a to an action representation $e \in \mathcal{E}$. The decoder $g^i_{\delta^i}$ parameterized by δ^i maps the action representation $e \in \mathcal{E}$ to the action probability. The processes are formulated as:

$$\textbf{Encoding}: e = f_{\phi^i}(s, s'), \forall s \in \mathcal{S}, \forall s' \in \mathcal{S}, \quad \textbf{Decoding}: a \sim g^i_{\delta^i}(\cdot|e), \forall e \in \mathcal{E}. \tag{2}$$

Although we use different superscripts to represent decoders in different tasks for clarity, the structure is continually updated rather than being task-specific. Therefore, $g^i_{\delta^i}$ needs to map action representation e to the action probability of any action space from past and current tasks during testing.

The probability of an action a given the state s and the next state s' can be represented as $g^i_{\delta^i}(a|f_{\phi^i}(s, s'))$. To measure the difference between the true action probability and the predicted action probability, we use the cross-entropy loss as the loss function of the encoder-decoder network:

$$\mathcal{L}(\phi^i, \delta^i) = -\sum_{(s,a,s')\in\mathcal{T}} \log P(a|s, s') = -\sum_{(s,a,s')\in\mathcal{T}} \log g^i_{\delta^i}(a|f_{\phi^i}(s, s')). \tag{3}$$

This loss function only depends on the environmental dynamic data which is reward-agnostic, the agent can build the action representation space $\mathcal{E}$ with low computational cost.

After the SSL process, the agent can use the learned action representation space to interact with the environment. The original policy $\pi^i : \mathcal{S} \times \mathcal{A} \to [0, 1]$ can be represented by another policy $\tilde{\pi}_{\theta^i} : \mathcal{S} \to \mathcal{E}$ and the decoder $g^i_{\delta^i} : \mathcal{E} \times \bigcup_{j=1}^{i} \mathcal{A}^j \to [0, 1]$:

$$\pi^i(a|s) = g^i_{\delta^i}(a|\tilde{\pi}_{\theta^i}(s)). \tag{4}$$

Then the policy can be trained by a standard RL algorithm to maximize the expected return:

$$J(\theta^i) = \mathbb{E}_{\tilde{\pi}_{\theta^i}}\left[\sum_{t=0}^{H^i} \gamma^t R(S_t, A_t, S_{t+1})\right]. \tag{5}$$

4.3 REGULARIZED FINE-TUNING

In the new task $i + 1$, the policy needs to generalize to the new action space $\mathcal{A}^{i+1}$. The structure of the decoder $g^{i+1}_{\delta^{i+1}}$ needs to be expanded or masked to adapt to the new action space. If the action space is expanded, that is $\mathcal{A}^{i+1} \supset \mathcal{A}^i$, the network is expanded by adding new neurons. The parameters of the old neurons are fixed and the parameters of new neurons are initialized randomly. If the action space is contracted, that is $\mathcal{A}^{i+1} \subset \mathcal{A}^i$, the output corresponding to the actions that are not in the new action space is masked. This strategy has been broadly studied in the works of architecture-based CL methods (Rusu et al., 2016; Mallya & Lazebnik, 2018; Mallya et al., 2018).

During training on the transitions of the new task, the decoder is fine-tuned with the regularization. We use the Elastic Weight Consolidation (EWC) to constrain the fine-tuning process, as it has been shown to be effective in mitigating the catastrophic forgetting in CL (Kirkpatrick et al., 2017). The encoder is also fine-tuned in the new tasks to continuously refine the action representation space $\mathcal{E}$. We do not impose constraints on the encoder because we have found that its plasticity is crucial for learning a good representation space. The loss function of the decoder network in Equation 3 is modified to include the EWC term:

$$\mathcal{L}(\phi^{i+1}, \delta^{i+1}) = -\sum_{(s,a,s')\in\mathcal{T}^{i+1}} \log g^{i+1}_{\delta^{i+1}}(a|f_{\phi^{i+1}}(s,s')) + \frac{\lambda}{2}\sum_{j=1}^{i}\sum_{k} F_k^j \left(\delta_k^{i+1} - \delta_k^j\right)^2, \tag{6}$$

where F_k^j is the k-th diagonal element of the Fisher information matrix of the parameters of the decoder network on task j, and λ is a regularization coefficient to balance the two terms. After the fine-tuning process, the agent can use the new decoder to interact with the environment. In order to maintain consistency with the standard pipeline of CRL, we still update the policy in the new action space. This process does not use any regularization, and the objective function is the same as Equation 5. In this way, the agent can achieve better performance in the new task with little additional computational cost. [The algorithm is provided in Appendix A.]

5 Experiments

5.1 Benchmark

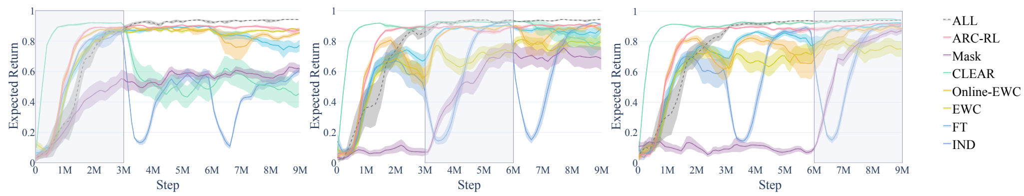
To evaluate the performance of CRL methods in SE-CRL, we establish a benchmark with changing action spaces. This benchmark comprises sequences with tasks that share identical state, reward, and transition dynamics but possess different action spaces. [Detailed descriptions of experimental settings, network structures, hyperparameters, and the metrics are provided in Appendix B.]

Environments. The environments of these tasks are based on MiniGrid (Chevalier-Boisvert et al., 2023) and Procgen (Cobbe et al., 2020). These environments feature image-based observations, a discrete set of possible actions. For expeditious training and evaluation, we use Empty of MiniGrid and Bigfish of Procgen in our experiments. Other environments are also provided in the Appendix C.7. The agents in these environments can move in different directions. To simulate the evolving capabilities, we introduce some additional actions and remove some existing actions. Then, we design three tasks with different numbers of actions for both MiniGrid and Procgen, specifically incorporating tasks with three, five, and seven actions for MiniGrid, and tasks with three, five, and nine actions for Bigfish. When the agent switches from one task to another, the set of available actions may either increase or decrease. The agent can only observe the action space of the current task. Based on these tasks, we evaluate the performance of CRL methods in three situations of task sequences: expansion (the action space is expanding), contraction (the action space is contracting), and the combination of both.

Compared Methods. We select three types of CRL methods to compare with SE-CRL: one replay-based method, **CLEAR** (Rolnick et al., 2019); two regularization-based methods, **EWC** (Kirkpatrick et al., 2017) and **online-EWC** (Schwarz et al., 2018); and one architecture-based method, **Mask** (Ben-Iwhiwhu et al., 2023). Additionally, we take the DRL methods trained with fine-tuning (named **FT**) and independently (named **IND**) across tasks as baselines. In the implementation of these methods, we adapt them to SE-CRL by using the largest action space of all tasks. This adaptation necessitates prior knowledge of all tasks, while our framework does not require it. In order to better understand the challenge of SE-CRL, we also introduce a baseline that is always able to access all action spaces (named **ALL**). This baseline does not involve CL and its final performance can be regarded as an upper bound of other methods. The underlying RL algorithm of all methods is IMPALA (Espeholt et al., 2018).

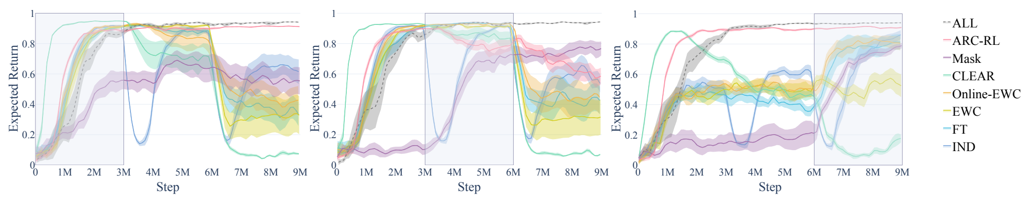
Metrics. To evaluate the effectiveness of ARC-RL, we use the expected return to measure the performance of the trained agents. Following the standard practice of CL (Díaz-Rodríguez et al., 2018; Wolczyk et al., 2021; Li et al., 2024b), we use three metrics based on the agent's performance throughout different phases of its training process: **continual return**, **forgetting**, and **forward transfer** (Powers et al., 2022). The continual return is the average performance achieved by the agent on all tasks after completing all training, which is consistent with the agent's objective in

Equation 1. The forgetting compares the expected return achieved for the earlier task before and after training on a new task, while the forward transfer compares the expected return achieved for the later task before and after training on an earlier task. Furthermore, the forward transfer metric measures the zero-shot generalization of the policy in SE-CRL. The averages of forgetting and forward transfer across all tasks are reported in the results.



(a) Task 1: Three actions (b) Task 2: Five actions (c) Task 3: Seven actions

Figure 4: Performance of eight methods on three MiniGrid tasks in the **expansion** situation.



(a) Task1: Seven actions (b) Task2: Five actions (c) Task3: Three actions

Figure 5: Performance of eight methods on three MiniGrid tasks in the **contraction** situation.

5.2 Competitive Experiments

To evaluate the effectiveness of our framework, we first compare it with other methods on MiniGrid tasks. Due to the simple environments, we only use a random policy in the exploration stage of ARC-RL. Each task is trained for 3M steps and replicated with 10 random seeds to ensure statistical reliability. During each task's training phase, the agent is not only trained on the current task but also periodically evaluated on all tasks, including those it has previously encountered. This evaluation allows us to assess both the learning progress on the current task and the retention of knowledge from prior tasks. The results presented in the evaluation plots and the total evaluation metric reported in the table below were computed as the mean of runs per method, with the shaded area and errors denoting the 95% confidence interval. Each subplot in the evaluation plots depicts the expected return of the agent evaluated on the corresponding task during training on all tasks, with the x-axis representing the total number of training steps across all tasks. The blue-shaded rectangular area indicates the training phase of the current task. We employ exponential moving averages to smooth the results for better visualization. As tasks are learned independently of other tasks in IND, there is no notion of forward transfer. Therefore, this method is omitted when forward transfer is reported. [Further details and more experiments (combined situations, longer sequences, and hyperparameter sensitivity analysis, etc) are provided in Appendix C]

Overall Performance. Figures 4 and 5 show the evaluated performance of eight methods on MiniGrid tasks with action spaces that are either expanding or contracting, respectively. The return curve of SE-CRL (red line) is generally higher than that of other methods across all tasks, suggesting that SE-CRL adapts more effectively to changing action spaces and achieves superior performance. Furthermore, the smaller shaded areas around the ARC-RL's curve also indicate greater stability compared to other methods. It is worth noting that although CLEAR can learn better in the first training phase, it experiences significant performance degradation on most tasks. This phenomenon indicates that the challenge of catastrophic forgetting in SE-CRL is different from existing CRL, and replay-based methods may not be able to effectively address it.

In the expansion situation, the overall performance of some methods slightly improves as training progresses (more evident in Figure 15). This phenomenon suggests that an expanding action space may facilitate policy generalization, echoing the principle of curriculum learning (Wang et al., 2022).

Table 1: Continual learning metrics of eight methods and two variants of ARC-RL across three MiniGrid tasks in situations of **expansion** and **contraction**. The average continual return of **ALL** is 0.94, which is not provided in the table. Continual return and forward transfer are abbreviated as “Return” and “Transfer”, respectively. The top five results are highlighted in green, and the depth of the color indicates the ranking.

Methods	Expansion			Contraction		
	Return↑	Forgetting↓	Transfer↑	Return↑	Forgetting↓	Transfer↑
IND	0.81 ± 0.02	0.08 ± 0.02	–	0.67 ± 0.06	0.26 ± 0.05	–
FT	0.86 ± 0.03	0.03 ± 0.02	0.48 ± 0.03	0.52 ± 0.07	0.39 ± 0.06	0.39 ± 0.03
EWC	0.81 ± 0.04	-0.04 ± 0.01	0.43 ± 0.05	0.39 ± 0.11	0.40 ± 0.09	0.47 ± 0.03
online-EWC	0.87 ± 0.02	0.02 ± 0.01	0.40 ± 0.05	0.56 ± 0.09	0.34 ± 0.06	0.44 ± 0.03
Mask	0.72 ± 0.05	-0.04 ± 0.04	0.02 ± 0.03	0.70 ± 0.06	-0.02 ± 0.04	0.08 ± 0.03
CLEAR	0.73 ± 0.06	0.21 ± 0.06	0.58 ± 0.01	0.11 ± 0.02	0.58 ± 0.03	0.46 ± 0.02
ARC-RL	0.90 ± 0.01	-0.02 ± 0.01	0.57 ± 0.02	0.80 ± 0.03	0.04 ± 0.03	0.60 ± 0.01
ARC-RL-O	0.86 ± 0.03	0.00 ± 0.03	0.51 ± 0.02	0.73 ± 0.06	0.04 ± 0.03	0.60 ± 0.01
ARC-RL-E	0.89 ± 0.03	-0.03 ± 0.03	0.55 ± 0.04	0.74 ± 0.05	0.18 ± 0.04	0.51 ± 0.03

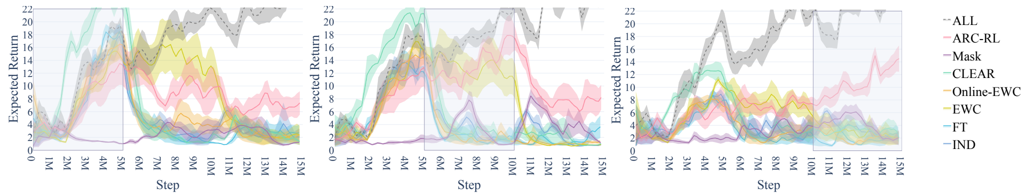
However, some methods experience a performance drop after training task changes (e.g., step 3M in Figure 4b), highlighting the challenge of policy generalization across different action spaces. ARC-RL, with minimal performance fluctuations upon action space changes, demonstrates the utility of the action representation space for policy learning and generalization.

In the contraction situation, performance changes are more pronounced. Most methods suffer significant performance shifts when the action space is reduced, indicating that policies trained on larger action spaces may not transfer well to smaller ones. This phenomenon further emphasizes the challenge of policy generalization across different action spaces. Although ARC-RL sometimes experiences a larger performance drop compared to Mask, which focuses on mitigating catastrophic forgetting, it generally outperforms other methods. After training on the final task with a three-action space, ARC-RL outshines others, demonstrating the benefits of action representation space for learning on new action spaces.

Continual Learning Performance. Table 1 presents the evaluation results in terms of CL metrics. The continual return metric demonstrates ARC-RL’s superiority in SE-CRL, significantly outperforming the other methods in all situations. The forward transfer metric is particularly noteworthy, as it measures the agent’s ability to leverage knowledge from previous tasks and indicates zero-shot generalization to new action spaces. ARC-RL exhibits the highest forward transfer underscoring the benefit of the action representation space for generalization. The forgetting metric of all methods is relatively high in the contraction situation, further underscoring the policy generalizability challenge in SE-CRL. When some actions are removed, the optimal policy may change significantly, leading to a performance drop. Note that regularization-based methods (EWC and online-EWC) can not mitigate catastrophic forgetting in this situation, possibly due to the large difference in networks’ parameters between different action spaces. Although ARC-RL does not achieve the best score for the forgetting metric, its exceptional forward transfer capabilities and strong average performance accentuate its proficiency in handling SE-CRL. The above findings highlight ARC-RL’s potential to markedly enhance the adaptability and generalization ability of reinforcement learning agents, positioning it as a highly viable solution for SE-CRL.

5.3 Ablation Study

We conduct an ablation study on MiniGrid tasks to investigate what affects ARC-RL’s performance in SE-CRL. We consider two variants: ARC-RL-O, which omits the regularization during fine-tuning, and ARC-RL-E, which uses the same regularization for the encoder and decoder. The results, presented in Table 1, reveal that ARC-RL-O exhibits better forward transfer than other methods, demonstrating that the action representation space is helpful for a more generalized policy. The comparison between ARC-RL and ARC-RL-O in forgetting and forward transfer suggests that regularization may improve policy stability. Nevertheless, ARC-RL’s superior continual return demonstrates that this balance is beneficial for the stability-plasticity trade-off essential in continual learning systems (Wang et al. 2024). Compared with ARC-RL, additional regularization in ARC-RL-E damages the forward transfer but does not mitigate forgetting. This may indicate that the plasticity of the encoder is essential for the learning of the action representation space.



(a) Task 1: Nine actions (b) Task 2: Five actions (c) Task 3: Three actions

Figure 6: Performance of eight methods on three Procgen tasks in the **contraction** situation.

5.4 More Challenging Experiments

To further evaluate the effectiveness of ARC-RL, we conduct experiments on the Bigfish environment from Procgen. These tasks are more challenging than MiniGrid tasks due to their larger state space and more complex control logic. To better extract features from images, all methods' networks were equipped with the IMPALA architecture (Espeholt et al., 2018). As demonstrated in previous experiments, the contraction situation better highlights the challenges of SE-CRL. Therefore, we focus on the contraction situation in these experiments. Each experiment is trained for 5M steps and replicated with 5 random seeds to ensure statistical reliability. Other experimental configurations are consistent with those used in the MiniGrid experiments.

Table 2: Continual learning metrics of seven methods across three Bigfish tasks in the **contraction** situation. The average continual return of **ALL** is 24.77, which is not provided in the table. The top three results are highlighted in green, and the depth of the color indicates the ranking.

Methods	Metrics		
	Return↑	Forgetting↓	Transfer↑
IND	1.66 ± 0.96	0.18 ± 0.02	–
FT	3.01 ± 1.39	0.14 ± 0.08	0.23 ± 0.02
EWC	1.74 ± 1.04	0.26 ± 0.06	0.16 ± 0.06
online-EWC	1.84 ± 0.80	0.14 ± 0.03	0.18 ± 0.07
Mask	1.49 ± 0.71	-0.01 ± 0.01	0.06 ± 0.06
CLEAR	1.48 ± 0.44	0.23 ± 0.02	0.11 ± 0.04
ARC-RL	10.03 ± 1.94	0.12 ± 0.07	0.19 ± 0.05

Figure 6 and Table 2 present the performance and metrics of seven methods across three Bigfish tasks, respectively. The performance gap between ALL and other methods is more pronounced in these experiments, indicating the challenges posed by the Bigfish environment. Consistent with the MiniGrid experiments, ARC-RL outperforms other methods across all tasks. Many methods experience significant performance drops after training task changes, but ARC-RL has a less volatile return curve, indicating effective policy generalization across different action spaces. Our method achieves strong results in both forgetting and positive transfer metrics. While some methods excel in one of these metrics, they fail to balance both simultaneously. The continual return, a crucial metric for CRL agents, varies significantly among different methods in this challenging task sequence. Popular CRL methods exhibit low returns after training on all tasks, likely due to suffering from both catastrophic forgetting and plasticity loss (Abbas et al., 2023). Interestingly, FT, a naive knowledge transfer method, performs better than other popular CRL methods, highlighting the distinct challenges posed by SE-CRL compared to existing CRL. Additionally, we also conducted experiments in the expansion situation, as detailed in Appendix C.6. Our proposed method also achieves optimal continual return in this setting, demonstrating its robustness across different task complexities. The experimental results further indicate that the agent effectively leverages the available actions to improve its policy. In summary, our method strikes a good balance between plasticity and stability, significantly outperforming other methods in terms of continual return.

6 Conclusion

In this paper, we first propose a new and practical problem to supplement CRL called *Self-Evolution Continual Reinforcement Learning* (SE-CRL), in which the agent's action space continuously changes. To tackle the challenges in this problem, we introduce a new framework called **A**ction **R**epresentation **C**ontinual **R**einforcement **L**earning (ARC-RL). This framework leverages self-supervised learning and regularized fine-tuning to build an action representation space to generalize the policy across different action spaces. We release a benchmark based on MiniGrid and Procgen to validate the effectiveness of CRL methods in SE-CRL. Experimental results demonstrate the superior performance of ARC-RL compared to popular CRL methods.

References

Zaheer Abbas, Rosie Zhao, Joseph Modayil, Adam White, and Marlos C. Machado. Loss of plasticity in continual deep reinforcement learning. In *CoLLAs*, volume 232, pp. 620–636, 2023.

David Abel, Yuu Jinnai, Sophie Yue Guo, George Konidaris, and Michael Littman. Policy and value transfer in lifelong reinforcement learning. In *ICML*, volume 80, pp. 20–29. PMLR, 10–15 Jul 2018.

Eseoghene Ben-Iwhiwhu, Saptarshi Nath, Praveen Kumar Pilly, Soheil Kolouri, and Andrea Soltoggio. Lifelong reinforcement learning with modulating masks. *Transactions on Machine Learning Research*, 2023.

Glen Berseth, Zhiwei Zhang, Grace Zhang, Chelsea Finn, and Sergey Levine. CoMPS: Continual meta policy search. In *ICLR*, 2022.

Douglas J Blackiston, Tal Shomrat, and Michael Levin. The stability of memories during brain remodeling: A perspective. *Communicative & Integrative Biology*, 8(5):e1073424, 2015.

Massimo Caccia, Jonas Mueller, Taesup Kim, Laurent Charlin, and Rasool Fakoor. Task-agnostic continual reinforcement learning: Gaining insights and overcoming challenges. In *CoLLAs*, volume 232, pp. 89–119. PMLR, 22–25 Aug 2023.

Yash Chandak, Georgios Theocharous, James Kostas, Scott Jordan, and Philip Thomas. Learning action representations for reinforcement learning. In *ICML*, volume 97, pp. 941–950, 2019.

Yash Chandak, Georgios Theocharous, Chris Nota, and Philip Thomas. Lifelong learning with a changing action set. In *AAAI*, volume 34, pp. 3373–3380, 2020.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and J Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. In *NeurIPS*, volume 36, pp. 73383–73394, 2023.

Mark M. Churchland, John P. Cunningham, Matthew T. Kaufman, Justin D. Foster, Paul Nuyujukian, Stephen I. Ryu, and Krishna V. Shenoy. Neural population dynamics during reaching. *Nature*, 487(7405):51–56, 2012.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *ICML*, volume 119, pp. 2048–2056, 2020.

Natalia Díaz-Rodríguez, Vincenzo Lomonaco, David Filliat, and Davide Maltoni. Don't forget, there is more than forgetting: New metrics for continual learning. *arXiv preprint arXiv:1810.13166*, 2018.

Wei Ding, Siyang Jiang, Hsi-Wen Chen, and Ming-Syan Chen. Incremental reinforcement learning with dual-adaptive ϵ-greedy exploration. In *AAAI*, volume 37, pp. 7387–7395, 2023.

Anila M. D'Mello, John D.E. Gabrieli, and Derek Evan Nee. Evidence for hierarchical cognitive control in the human cerebellum. *Current Biology*, 30(10):1881–1892.e3, 2020.

Maya Emmons-Bell, Fallon Durant, Angela Tung, Alexis Pietak, Kelsie Miller, Anna Kane, Christopher J Martyniuk, Devon Davidian, Junji Morokuma, and Michael Levin. Regenerative adaptation to electrochemical perturbation in planaria: A molecular analysis of physiological plasticity. *iScience*, 22:147–165, 2019.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *ICML*, volume 80, pp. 1407–1416, 2018.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In *NeurIPS*, volume 35, pp. 35603–35620. Curran Associates, Inc., 2022.

Ching Fang and Kim Stachenfeld. Predictive auxiliary objectives in deep RL mimic learning in the brain. In *ICLR*, 2024.

Naomi P Friedman and Trevor W Robbins. The role of prefrontal cortex in cognitive control and executive function. *Neuropsychopharmacology*, 47(1):72–89, 2022.

Juan A. Gallego, Matthew G. Perich, Raeed H. Chowdhury, Sara A. Solla, and Lee E. Miller. Long-term stability of cortical population dynamics underlying consistent behavior. *Nature Neuroscience*, 23(2):260–270, 2020.

Jean-Baptiste Gaya, Thang Doan, Lucas Caccia, Laure Soulier, Ludovic Denoyer, and Roberta Raileanu. Building a subspace of policies for scalable continual learning. In *NeurIPS DRL Workshop*, 2022.

Michael S Gazzaniga, Richard B Ivry, and GR Mangun. *Cognitive Neuroscience. The Biology of the Mind*. W.W. Norton & Company, New York, 2019. ISBN 978-7-5184-4043-6.

Apostolos P. Georgopoulos and Giuseppe Pellizzer. The mental and the neural: Psychological and neural studies of mental rotation and memory scanning. *Neuropsychologia*, 33(11):1531–1547, 1995.

Shlomi Haar, Opher Donchin, and Ilan Dinstein. Dissociating visual and motor directional selectivity using visuomotor adaptation. *Journal of Neuroscience*, 35(17):6813–6821, 2015.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *ICLR*, 2020.

Shinji Kakei, Donna S. Hoffman, and Peter L. Strick. Muscle and movement representations in the primary motor cortex. *Science*, 285(5436):2136–2139, 1999.

Christos Kaplanis, Murray Shanahan, and Claudia Clopath. Policy consolidation for continual reinforcement learning. In *ICML*, volume 97, pp. 3242–3251, 2019.

Samuel Kessler, Jack Parker-Holder, Philip Ball, Stefan Zohren, and Stephen J. Roberts. Same state, different task: Continual reinforcement learning without interference. In *AAAI*, volume 36, pp. 7143–7151, 2022.

Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *Journal of Academia and Industrial Research*, 75: 1401–1476, 2022.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Sam Kriegman, Stephanie Walker, Dylan Shah, Michael Levin, Rebecca Kramer-Bottiglio, and Josh Bongard. Automated shapeshifting for function recovery in damaged robots. In *RSS*, 2019.

Dhireesha Kudithipudi, Mario Aguilar-Simon, Jonathan Babb, Maxim Bazhenov, Douglas Blackiston, Josh Bongard, Andrew P. Brna, Suraj Chakravarthi Raja, Nick Cheney, Jeff Clune, Anurag Daram, Stefano Fusi, Peter Helfer, Leslie Kay, Nicholas Ketz, Zsolt Kira, Soheil Kolouri, Jeffrey L. Krichmar, Sam Kriegman, Michael Levin, Sandeep Madireddy, Santosh Manicka, Ali Marjaninejad, Bruce McNaughton, Risto Miikkulainen, Zaneta Navratilova, Tej Pandit, Alice Parker, Praveen K. Pilly, Sebastian Risi, Terrence J. Sejnowski, Andrea Soltoggio, Nicholas Soures, Andreas S. Tolias, Darío Urbina-Meléndez, Francisco J. Valero-Cuevas, Gido M. van de Ven, Joshua T. Vogelstein, Felix Wang, Ron Weiss, Angel Yanguas-Gil, Xinyun Zou, and Hava Siegelmann. Biological underpinnings for lifelong learning machines. *Nature Machine Intelligence*, 4(3):196–210, 2022.

Robert Kwiatkowski and Hod Lipson. Task-agnostic self-modeling machines. *Science Robotics*, 4 (26):eaau9354, 2019.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In *ICML*, volume 119, pp. 5639–5650. PMLR, 13–18 Jul 2020a.

Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. In *NeurIPS*, volume 33, pp. 19884–19895. Curran Associates, Inc., 2020b.

Xiang Li, Jinghuan Shang, Srijan Das, and Michael Ryoo. Does self-supervised learning really improve reinforcement learning from pixels? In *NeurIPS*, volume 35, pp. 30865–30881. Curran Associates, Inc., 2022.

Yanhua Li, Jiafen Liu, Longhao Yang, Chaofan Pan, Xiangkun Wang, and Xin Yang. Three-way open intent classification with nearest centroid-based representation. *Information Sciences*, 681: 121251, 2024a.

Yujie Li, Xin Yang, Hao Wang, Xiangkun Wang, and Tianrui Li. Learning to prompt knowledge transfer for open-world continual learning. In *AAAI*, volume 38, pp. 13700–13708, 2024b.

Jiashun Liu, Jianye HAO, Yi Ma, and Shuyin Xia. Unlock the cognitive generalization of deep reinforcement learning via granular ball representation. In *ICML*, 2024.

Laurens Van Der Maaten and Geoffrey Hinton. Visualizing data using T-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

Arun Mallya and Svetlana Lazebnik. PackNet: Adding multiple tasks to a single network by iterative pruning. In *CVPR*, pp. 7765–7773, 2018.

Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *ECCV*, pp. 67–82, 2018.

Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D. Bagdanov, and Joost van de Weijer. Class-incremental learning: Survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5513–5533, 2022.

Chaofan Pan, Xin Yang, Hao Wang, Wei Wei, and Tianrui Li. Hi-core: Hierarchical knowledge transfer for continual reinforcement learning. *arXiv preprint arXiv:2401.15098*, 2024.

Vitchyr Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. In *ICML*, volume 119, pp. 7783–7792. PMLR, 13–18 Jul 2020.

Sam Powers, Eliot Xing, Eric Kolve, Roozbeh Mottaghi, and Abhinav Gupta. CORA: Benchmarks, baselines, and metrics as a platform for continual reinforcement learning agents. In *CoLLAs*, volume 199, pp. 705–743, 2022.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *NeurIPS*, volume 32, 2019.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Bing Liu Sahisnu Mazumder. *Lifelong and Continual Learning Dialogue Systems*. Springer, 2024. ISBN 978-3-031-48188-8.

Philemon Schöpf, Sayantan Auddy, Jakob Hollenstein, and Antonio Rodriguez-sanchez. Hypernetwork-PPO for continual reinforcement learning. In *NeurIPS DRL Workshop*, 2022.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *ICML*, volume 80, pp. 4528–4537, 2018.

Jonathan Schwarz, Siddhant M. Jayakumar, Razvan Pascanu, Peter E. Latham, and Yee Whye Teh. Powerpropagation: A sparsity inducing weight reparameterisation. In *NeurIPS*, pp. 28889–28903. Curran Associates, Inc., 2021.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *NeurIPS*, volume 30, 2017.

Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. In *ICML*, volume 139, pp. 9870–9879. PMLR, 18–24 Jul 2021.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5362–5383, 2024a. doi: 10.1109/TPAMI.2024.3367329.

Xin Wang, Yudong Chen, and Wenwu Zhu. A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):4555–4576, 2022.

Xu Wang, Sen Wang, Xingxing Liang, Dawei Zhao, Jincai Huang, Xin Xu, Bin Dai, and Qiguang Miao. Deep reinforcement learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(4):5064–5078, 2024b. doi: 10.1109/TNNLS.2022.3207346.

Zhi Wang, Han-Xiong Li, and Chunlin Chen. Incremental reinforcement learning in continuous spaces via policy relaxation and importance weighting. *IEEE Transactions on Neural Networks and Learning Systems*, 31(6):1870–1883, 2019.

Matthew Weightman, Neeraj Lalji, Chin-Hsuan Sophie Lin, Joseph M. Galea, Ned Jenkinson, and R. Chris Miall. Short duration event related cerebellar tdcs enhances visuomotor adaptation. *Brain Stimulation*, 16(2):431–441, 2023.

Maciej Wolczyk, Michal Zajac, Razvan Pascanu, Lukasz Kucinski, and Piotr Milos. Continual World: A robotic benchmark for continual reinforcement learning. In *NeurIPS*, pp. 28496–28510, 2021.

Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *ICLR*, 2021.

Deheng Ye, Zhao Liu, Mingfei Sun, Bei Shi, Peilin Zhao, Hao Wu, Hongsheng Yu, Shaojie Yang, Xipeng Wu, Qingwei Guo, et al. Mastering complex control in moba games with deep reinforcement learning. In *AAAI*, volume 34, pp. 6672–6679, 2020.

William Yue, Bo Liu, and Peter Stone. t-DGR: A trajectory-based deep generative replay method for continual learning in decision making. In *NeurIPS ALOE Workshop*, 2023.

A Framework Details

Algorithm 1 shows the complete process of ARC-RL. The notations used in the algorithm are consistent with those in the main text. For each task, ARC-RL consists of two stages: exploration and learning. The parameters ϕ, δ, and θ are continually updated in place as the process. After all tasks are completed, the policy $\pi_{E\theta}$ and decoder g_δ of the final task are returned. Therefore, we do not use superscript i to denote the parameters in the algorithm.

Algorithm 1 ARC-RL

Input: Tasks with different action space $\{\mathcal{A}^i\}_{i=1}^N$.
Initialize: θ, ϕ and δ.
for $t = 1, 2, \ldots, M$ **do**
  See Task with action space $\mathcal{A}^i$

  Exploration Stage

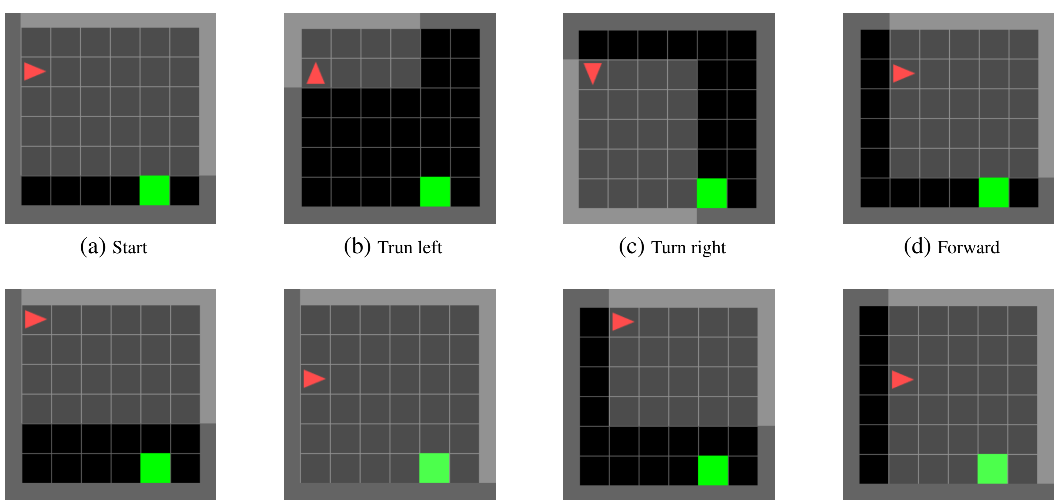
  Use exploration policy to interact with the environment to collect transitions $\mathcal{T}^i$;
  if $t = 1$ **then**
    Update ϕ and δ with by minimizing Equation 3; //Encoder-decoder training
  else
    Update ϕ and δ with by minimizing Equation 6; //Regularized fine-tuning

  Learning Stage

  Use Equation 4 to interact with the environment;
  Update θ by maximizing Equation 5; //Policy training

Return: Policy $\pi_{E\theta}$ and decoder g_δ.

B Environmental Details



(a) Start (b) Trun left (c) Turn right (d) Forward

(e) Left (f) Right (g) Forward left (h) Forward right

Figure 7: The screenshots of actions in MiniGrid. The transparent white area represents the agent's field of view, which is the state. (a) The agent starts from this state. (b)–(h) The agent's state after the corresponding action.

(a) (b) (c)

Figure 8: The screenshots of Bigfish. The texture and objects are procedurally generated.

B.1 Environments and Task Sequences.

MiniGrid[1] The MiniGrid contains a collection of simple 2D grid-world environments with a variety of objects, such as walls, doors, keys, and agents. These environments feature image-based partial observations, a discrete set of possible actions, and various objects characterized by their color and type. For expeditious training and evaluation, we only use the empty room environments of MiniGrid in our experiments. By default, they have a discrete 7-dimensional action space and produce a 3-channel integer state encoding of the 7×7 grid directly including and in front of the agent. Following the training setup for Atari (Schrittwieser et al., 2020), we modified the environments to output a $7 \times 7 \times 9$ by stacking three frames. Furthermore, we only use three basic movement actions from the original action space of MiniGrid: turn left, turn right, and move forward. Then, we expand the action space by adding four more actions to simulate SE-CRL: move left, move right, forward left, forward right. The screenshots of these actions are shown in Figure 7. Finally, we design three tasks with different action spaces: a three-action task (turn left, turn right, forward), a five-action task (turn left, turn right, forward, left, right), and a seven-action task (turn left, turn right, forward, left, right, forward left, forward right).

Procgen.[2] The Procgen benchmark is a collection of procedurally generated environments designed to evaluate generalization in RL algorithms. It was proposed as a replacement for the Atari games benchmark while being computationally faster to simulate than Atari. For faster training and evaluation, we chose Bigfish with the easiest level as the base environment in our experiments, in which the agent starts as a small fish and needs to become bigger by eating other fish. Figure 8 shows the screenshots of this environment. The input observations are RGB images of dimension $64 \times 64 \times 3$, along with 15 possible discrete actions. Similar to MiniGrid, we only use nine basic movement actions from the original action space of Bigfish: stay, up, down, left, right, up-left, up-right, down-left, and down-right. Then, we design three tasks with different action spaces: a three-action task (stay, up, down), a five-action task (stay, up, down, left, right), and a nine-action task (stay, up, down, left, right, up-left, up-right, down-left, down-right).

B.2 Compared Methods.

We compare ARC-RL with six methods in SE-CRL. These methods cover three common types of CRL: replay-based, regularization-based, and architecture-based. Note that CLEAR is task-agnostic, while EWC and ARC-RL require explicit task boundaries. The details of the methods are as follows:

- **IND.** This method represents a traditional DRL setup where an agent is trained independently on each task. This serves as a foundational comparison point to underscore the advantages of CL, as it lacks any mechanism for knowledge retention or transfer.
- **FT.** Building upon the standard DRL algorithm, this method differs from IND by using a single agent that is sequentially fine-tuned across different tasks. As a naive CRL method,

[1] `https://minigrid.farama.org/environments/minigrid/EmptyEnv/`
[2] `https://github.com/openai/procgen`

Table 3: Network structure for MiniGrid. All convolutional layers use a kernel size of 2×2 and a stride of 1. Linear 2 and Linear 4 are the output heads in ARC-RL, while Linear 3 and Linear 5 are the output heads in other methods.

	Layer	Input channels/units	Output channels/units
Backbone	Conv 1	9	32
	Conv 2	32	64
	Conv 3	64	128
	Linear 1	2048	64
Value output	Linear 2	64+256+1	1
	Linear 3	64+7+1	1
Policy output	Linear 4	64+256+1	256
	Linear 5	64+7+1	7
Encoder	Conv 4	9	32
	Conv 5	32	64
	Conv 6	64	128
	Linear 6	2048	64
	Linear 7	64	256
Decoder	Linear 8	256	7

this method provides a basic measure of an agent's capacity to maintain knowledge of earlier tasks while encountering new tasks (Gaya et al., 2022).

- **CLEAR.** A classical CRL method aiming to mitigate catastrophic forgetting by using a replay buffer to store experiences from previous tasks (Rolnick et al., 2019). It uses off-policy learning and behavioral cloning from replay to enhance stability, as well as on-policy learning to preserve plasticity.
- **EWC.** An RL implementation of Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017), which is designed to mitigate catastrophic forgetting by selectively constraining the update of weights that are important for previous tasks.
- **Online-EWC.** A modified version of EWC that adds an explicit forgetting mechanism to perform well with low computational cost (Schwarz et al., 2018).
- **Mask.** A CRL method that adapts modulating masks to the network architecture to prevent catastrophic forgetting (Ben-Iwhiwhu et al., 2023)[1]. The linear combination of the previously learned masks is used to exploit knowledge when learning new tasks.

B.3 Network Structures

All methods in our experiments are implemented based on IMPALA (Espeholt et al., 2018). The network of this algorithm is consistent across all methods, except for the specific components of each method. For MiniGrid, we use a small network as the input observation is an image with shape $9 \times 7 \times 7$. As shown in Table 3, each network consists of a convolutional neural network (CNN) with three convolutional layers and two fully connected layers. ReLU activation is employed in all networks except the output layers of the policy network in ARC-RL, which uses a sigmoid activation. Note that the number of input units for the policy and value output heads changes because the one-hot action vector and reward scalar from the previous time step are concatenated to the output of Linear 1. For Procgen, we replace the CNN with the IMPALA architecture to improve the representation ability of bigger images. As shown in Table 4, this architecture consists of three IMPALA blocks, each of which contains a convolutional layer and two residual blocks. Additionally, we employ a bigger CNN in the encoder of ARC-RL to extract features from the input image.

[1] We use the code at `https://github.com/dlpbc/mask-lrl-procgen/tree/develop_v2`

Table 4: Network structure for Procgen. All convolutional layers in the backbone use a kernel size of 3×3 and a stride of 1. The kernel sizes of the convolutional layers in the encoder are 8×8, 4×4, and 3×3, respectively. The stride of them are 4, 2, and 1, respectively. All maxpool layers use a kernel size of 3×3 and a stride of 2. Linear 2 and Linear 4 are the output heads in ARC-RL, while Linear 3 and Linear 5 are the output heads in other methods.

	Layer	Input channels/units	Output channels/units
Backbone	Conv 1	3	32
	MaxPool	32	32
	Residual 1	32	32
	Residual 2	32	32
	Conv 2	32	64
	MaxPool	64	64
	Residual 3	64	64
	Reedisidual 4	64	64
	Conv 3	64	64
	MaxPool	64	64
	Residual 5	64	64
	Reedisidual 6	64	64
	Linear 1	3136	512
Value output	Linear 2	512+256+1	1
	Linear 3	512+9+1	1
Policy output	Linear 4	512+256+1	256
	Linear 5	512+9+1	7
Encoder	Conv 4	9	32
	Conv 5	32	64
	Conv 6	64	64
	Linear 6	1024	512
	Linear 7	512	256
Decoder	Linear 8	256	7

B.4 HYPERPARAMETERS

The hyperparameters for the competitive experiments are presented in Table 5 and Table 6. Most values follow the settings in CORA (Powers et al., 2022). Note that for ARC-RL, we did not conduct experiments to search for the best hyperparameters. Additionally, the number of exploration steps for ARC-RL on new tasks is set to 10^4. This parameter is relatively small compared to the number of training steps per task and is not been tuned. In the implementation of CLEAR, each actor only gets sampled once during training, so we need the same number of actors as well as batch size.

B.5 METRICS

Based on the agent's normalized expected return, we evaluated the continual learning performance of our framework and other methods using the following metrics: continual return, forgetting, and forward transfer. Let us consider a sequence with N tasks, where $p_{i,j} \in [0, 1]$ represents the performance of task j (evaluation return) after the agent has been trained on task i. Then, the above metrics can be defined:

- **Continual return**: The continual return for task i is defined as:

$$\mathbf{R}_i := \frac{1}{i} \sum_{j=1}^{i} p_{i,j}. \tag{7}$$

  This metric provides an overall view of the agent's performance up to and including task i. The final value, $\mathbf{R} \in [0, 1]$ is a single-value summary of the agent's overall performance after all tasks and is included in the result tables.

- **Forgetting**: The forgetting for task i measures the decline in performance for that task after training has concluded. It is calculated by:

$$\mathbf{F}_i := \frac{1}{i-1} \sum_{j=1}^{i-1} (p_{i-1,j} - p_{i,j}) \tag{8}$$

Table 5: Hyperparameters for the experiments on MiniGrid. λ is the regularization coefficient.

Method	Hyperparameter	Value
Common	Num. of actors	6
	Num. of learner	2
	Batch size	256
	Learning rate	4×10^{-4}
	Entropy	0.01
	Rollout length	20
	Optimizer	RMSProp
	Discount factor	0.99
	Gradient clip	40
	Num. of training steps per task	3×10^6
	Num. of evaluation episodes	10
	Evaluation interval	10^5
CLEAR	Num. of actors	12
	Batch size	12
	Replay buffer size	5×10^6
EWC	λ	10^4
	Replay buffer size	10^6
	Min. frames per task	2×10^5
Online-EWC	λ	175
	Replay buffer size	10^6
P&C	λ	3000
	Replay buffer size	10^5
	Num. of progress train steps	3906
ARC-RL	λ	2×10^4
	Action Representation size	256

Table 6: Hyperparameters for the experiments on Procgen. Other hyperparameters are the same as those on MiniGrid.

Hyperparameter	Value
Num. of actors	21
Num. of learner	2
Batch size	32
Num. of training steps per task	5×10^6
Num. of evaluation episodes	10
Evaluation interval	2.5×10^5

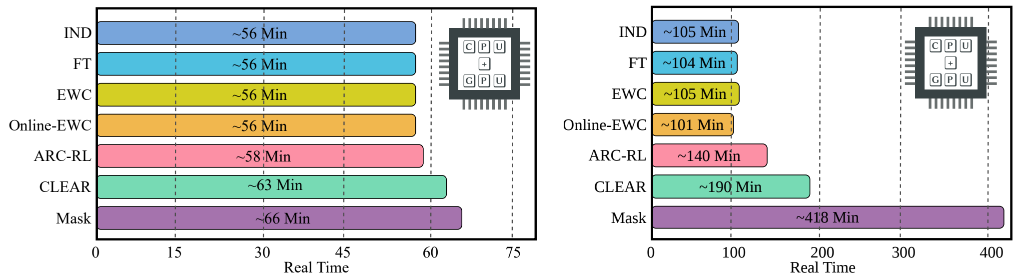
When $\mathbf{F}_i > 0$, the agent has become worse at the past tasks after training on new task i, indicating forgetting has occurred. Conversely, when $\mathbf{F}_i < 0$, the agent has become better at past tasks, indicating backward transfer has been observed. The overall forgetting metric, $\mathbf{F} \in [-1, 1]$, is the average of $\mathbf{F}_i$ values for all tasks, providing insight into how much knowledge the agent retains over time. We report $\mathbf{F}$ in the results tables.

- **Forward transfer**: The forward transfer for task i quantifies the positive impact that learning task i has on the performance of subsequent tasks. It is computed as follows:

$$\mathbf{T}_i := \frac{1}{N-i} \sum_{j=i+1}^{N} (p_{i,j} - p_{i-1,j}) \tag{9}$$

When $\mathbf{T}_i > 0$, the agent has become better at later tasks after training on earlier task i, indicating forward transfer has occurred through zero-shot learning. When $\mathbf{T}_i < 0$, the agent has become worse at later tasks, indicating negative transfer has occurred. The overall forward transfer, $\mathbf{T} \in [-1, 1]$, is the mean of $\mathbf{T}_i$ values across all tasks, providing insight into the generalization ability of the agent. We report $\mathbf{T}$ in the results tables.

B.6 Compute resources



(a) MiniGrid tasks

(b) Procgen tasks

Figure 9: Average runtime of seven methods on three MiniGrid tasks and three Procgen tasks.

Our code is implemented with Python 3.9.17 and Torch 2.0.1+cu118. Each method on MiniGrid was trained using an AMD Ryzen 5 3600 CPU (6 cores) with 32GB RAM and an NVIDIA GeForce RTX 1060 GPU. The Procgen experiments were conducted on an AMD Ryzen 9 7950X CPU (16 cores) with 48GB RAM and an NVIDIA GeForce RTX 4070Ti Super GPU. As illustrated in Figure 9a, each run, consisting of three MiniGrid tasks, takes about 1 hour to complete. However, there is a notable difference in the runtime of the methods when applied to Procgen tasks, as shown in Figure 9b. Specifically, the CLEAR and Mask take approximately twice and four times as long as the baselines, respectively. This increased runtime may attributed to the additional computation required to update the replay buffer and masks. Although the runtime of ARC-RL is longer than that of the baselines, it remains acceptable for practical applications when compared to CLEAR and Mask. In summary, the total runtime is influenced by four factors: the device, the domain, the computation time of the algorithm, and the behavior of the policy. Replay-based and architecture-based methods may experience a sharp increase in runtime due to heightened task complexity. In contrast, our method requires only a modest amount of additional computing resources to explore the environment, thereby achieving a balanced trade-off between efficiency and effectiveness.

C Additional Experiments and Results

C.1 Detailed Results on MiniGrid

Tables 7 and 8 present the detailed results of forgetting and forward transfer across three MiniGrid tasks in the situations of expansion and contraction. The columns represent trained tasks, while the rows represent evaluated tasks. We denote the task with an action space of size n as "n-Actions". The average results across all tasks (bottom right of each subtable) are reported in the main text. In each forgetting table, negative values are shown in green and positive values in red, with darker shades representing larger magnitudes. Values close to zero are unshaded. In each forward transfer table, positive values are shown in green, and negative values in red.

These results further highlight the superiority of ARC-RL. Additionally, they illustrate the difference between various action spaces and situations. Forgetting is slight in the expansion situation but more pronounced in the contraction situation. After training with 3-actions, the performance on previous tasks significantly degrades. However, forward transfer remains similar across different situations. In the expansion situation, training on 3-actions benefits evaluation performance on the subsequent tasks. In the contraction situation, training on 7-actions similarly benefits performance on subsequent tasks. This phenomenon may be attributed to the high similarity between tasks, where learning on the first task aids in learning subsequent tasks.

C.2 Combined Situations

Figures 10 and 11, along with Table 9, show the performance of SE-CRL and other CRL methods in combined situations of expansion and contraction across three MiniGrid tasks. We use "expansion & contraction" to represent the situation where the size of action space expands to seven after the first

Table 7: Forgetting and forward transfer in the **expansion** situation across three MiniGrid tasks.

(a) IND

	Forgetting			
	3-Actions	5-Actions	7-Actions	Avg ± SEM
3-Actions	–	0.31 ± 0.05	-0.05 ± 0.07	0.13 ± 0.02
5-Actions	–	–	-0.03 ± 0.02	-0.03 ± 0.02
7-Actions	–	–	–	–
Avg ± SEM	–	0.31 ± 0.05	-0.04 ± 0.04	0.08 ± 0.02

(b) FT

	Forgetting				Forward Transfer			
	3-Actions	5-Actions	7-Actions	Avg ± SEM	3-Actions	5-Actions	7-Actions	Avg ± SEM
3-Actions	–	-0.03 ± 0.02	0.12 ± 0.05	0.05 ± 0.03	–	–	–	–
5-Actions	–	–	0.01 ± 0.02	0.01 ± 0.02	0.69 ± 0.05	–	–	0.69 ± 0.05
7-Actions	–	–	–	–	0.59 ± 0.07	0.15 ± 0.09	–	0.37 ± 0.04
Avg ± SEM	–	-0.03 ± 0.02	0.07 ± 0.03	0.03 ± 0.02	0.64 ± 0.04	0.15 ± 0.09	–	0.48 ± 0.03

(c) EWC

	Forgetting				Forward Transfer			
	3-Actions	5-Actions	7-Actions	Avg ± SEM	3-Actions	5-Actions	7-Actions	Avg ± SEM
3-Actions	–	-0.04 ± 0.02	-0.02 ± 0.01	-0.03 ± 0.01	–	–	–	–
5-Actions	–	–	-0.06 ± 0.03	-0.06 ± 0.03	0.55 ± 0.09	–	–	0.55 ± 0.09
7-Actions	–	–	–	–	0.64 ± 0.06	0.09 ± 0.09	–	0.36 ± 0.04
Avg ± SEM	–	-0.04 ± 0.02	-0.04 ± 0.01	-0.04 ± 0.01	0.60 ± 0.07	0.09 ± 0.09	–	0.43 ± 0.05

(d) Online-EWC

	Forgetting				Forward Transfer			
	3-Actions	5-Actions	7-Actions	Avg ± SEM	3-Actions	5-Actions	7-Actions	Avg ± SEM
3-Actions	–	-0.03 ± 0.04	0.04 ± 0.03	0.01 ± 0.02	–	–	–	–
5-Actions	–	–	0.03 ± 0.03	0.03 ± 0.03	0.50 ± 0.09	–	–	0.50 ± 0.09
7-Actions	–	–	–	–	0.58 ± 0.09	0.12 ± 0.11	–	0.35 ± 0.05
Avg ± SEM	–	-0.03 ± 0.04	0.04 ± 0.02	0.02 ± 0.01	0.54 ± 0.08	0.12 ± 0.11	–	0.40 ± 0.05

(e) CLEAR

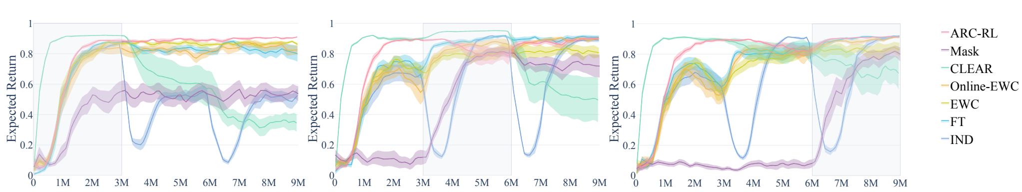
	Forgetting				Forward Transfer			
	3-Actions	5-Actions	7-Actions	Avg ± SEM	3-Actions	5-Actions	7-Actions	Avg ± SEM
3-Actions	–	0.44 ± 0.09	0.03 ± 0.13	0.24 ± 0.05	–	–	–	–
5-Actions	–	–	0.15 ± 0.07	0.15 ± 0.07	0.87 ± 0.03	–	–	0.87 ± 0.03
7-Actions	–	–	–	–	0.87 ± 0.02	-0.01 ± 0.01	–	0.43 ± 0.01
Avg ± SEM	–	0.44 ± 0.09	0.09 ± 0.09	0.21 ± 0.06	0.87 ± 0.02	-0.01 ± 0.01	–	0.58 ± 0.01

(f) MASK

	Forgetting				Forward Transfer			
	3-Actions	5-Actions	7-Actions	Avg ± SEM	3-Actions	5-Actions	7-Actions	Avg ± SEM
3-Actions	–	0.01 ± 0.11	-0.15 ± 0.07	-0.07 ± 0.05	–	–	–	–
5-Actions	–	–	0.02 ± 0.07	0.02 ± 0.07	0.03 ± 0.03	–	–	0.03 ± 0.03
7-Actions	–	–	–	–	-0.04 ± 0.05	0.08 ± 0.03	–	0.02 ± 0.03
Avg ± SEM	–	0.01 ± 0.11	-0.06 ± 0.06	-0.04 ± 0.04	-0.00 ± 0.03	0.08 ± 0.03	–	0.02 ± 0.03

(g) ARC-RL

	Forgetting				Forward Transfer			
	3-Actions	5-Actions	7-Actions	Avg ± SEM	3-Actions	5-Actions	7-Actions	Avg ± SEM
3-Actions	–	-0.01 ± 0.02	-0.01 ± 0.02	-0.01 ± 0.01	–	–	–	–
5-Actions	–	–	-0.04 ± 0.04	-0.04 ± 0.04	0.88 ± 0.03	–	–	0.88 ± 0.03
7-Actions	–	–	–	–	0.89 ± 0.04	-0.05 ± 0.02	–	0.42 ± 0.02
Avg ± SEM	–	-0.01 ± 0.02	-0.02 ± 0.02	-0.02 ± 0.01	0.88 ± 0.03	-0.05 ± 0.02	–	0.57 ± 0.02



(a) Task 1: Three actions (b) Task 2: Seven actions (c) Task 3: Five actions

Figure 10: Performance of seven methods on three MiniGrid tasks in the expansion & contraction situation.

Table 8: Forgetting and forward transfer in the **contraction** situation across three MiniGrid tasks.

(a) IND

	Forgetting			
	7-Actions	5-Actions	3-Actions	Avg ± SEM
7-Actions	–	0.06 ± 0.02	0.25 ± 0.09	0.15 ± 0.04
5-Actions	–	–	0.48 ± 0.08	0.48 ± 0.08
3-Actions	–	–	–	–
Avg ± SEM	–	0.06 ± 0.02	0.37 ± 0.08	0.26 ± 0.05

(b) FT

	Forgetting				Forward Transfer			
	7-Actions	5-Actions	3-Actions	Avg ± SEM	7-Actions	5-Actions	3-Actions	Avg ± SEM
7-Actions	–	0.01 ± 0.01	0.60 ± 0.10	0.31 ± 0.05	–	–	–	–
5-Actions	–	–	0.56 ± 0.10	0.56 ± 0.10	0.82 ± 0.04	–	–	0.82 ± 0.04
3-Actions	–	–	–	–	0.42 ± 0.07	-0.07 ± 0.08	–	0.18 ± 0.03
Avg ± SEM	–	0.01 ± 0.01	0.58 ± 0.09	0.39 ± 0.06	0.62 ± 0.05	-0.07 ± 0.08	–	0.39 ± 0.03

(c) EWC

	Forgetting				Forward Transfer			
	7-Actions	5-Actions	3-Actions	Avg ± SEM	7-Actions	5-Actions	3-Actions	Avg ± SEM
7-Actions	–	-0.01 ± 0.02	0.63 ± 0.13	0.31 ± 0.07	–	–	–	–
5-Actions	–	–	0.59 ± 0.12	0.59 ± 0.12	0.88 ± 0.03	–	–	0.88 ± 0.03
3-Actions	–	–	–	–	0.54 ± 0.07	-0.02 ± 0.06	–	0.26 ± 0.03
Avg ± SEM	–	-0.01 ± 0.02	0.61 ± 0.12	0.40 ± 0.09	0.71 ± 0.04	-0.02 ± 0.06	–	0.47 ± 0.03

(d) Online-EWC

	Forgetting				Forward Transfer			
	7-Actions	5-Actions	3-Actions	Avg ± SEM	7-Actions	5-Actions	3-Actions	Avg ± SEM
7-Actions	–	0.06 ± 0.05	0.45 ± 0.10	0.26 ± 0.05	–	–	–	–
5-Actions	–	–	0.51 ± 0.12	0.51 ± 0.12	0.85 ± 0.05	–	–	0.85 ± 0.05
3-Actions	–	–	–	–	0.47 ± 0.03	-0.00 ± 0.06	–	0.23 ± 0.03
Avg ± SEM	–	0.06 ± 0.05	0.48 ± 0.10	0.34 ± 0.06	0.66 ± 0.04	-0.00 ± 0.06	–	0.44 ± 0.03

(e) CLEAR

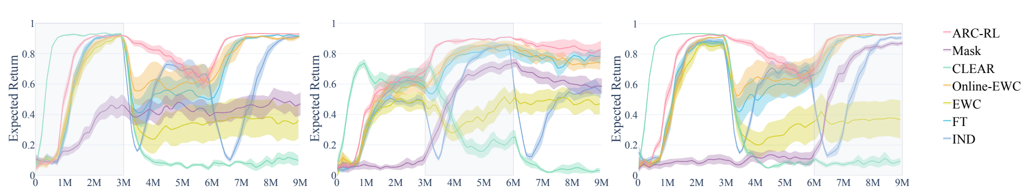
	Forgetting				Forward Transfer			
	7-Actions	5-Actions	3-Actions	Avg ± SEM	7-Actions	5-Actions	3-Actions	Avg ± SEM
7-Actions	–	0.29 ± 0.14	0.66 ± 0.15	0.47 ± 0.01	–	–	–	–
5-Actions	–	–	0.80 ± 0.07	0.80 ± 0.07	0.85 ± 0.03	–	–	0.85 ± 0.03
3-Actions	–	–	–	–	0.64 ± 0.05	-0.12 ± 0.08	–	0.26 ± 0.03
Avg ± SEM	–	0.29 ± 0.14	0.73 ± 0.11	0.58 ± 0.03	0.75 ± 0.03	-0.12 ± 0.08	–	0.46 ± 0.02

(f) MASK

	Forgetting				Forward Transfer			
	7-Actions	5-Actions	3-Actions	Avg ± SEM	7-Actions	5-Actions	3-Actions	Avg ± SEM
7-Actions	–	-0.08 ± 0.12	0.05 ± 0.09	-0.02 ± 0.06	–	–	–	–
5-Actions	–	–	-0.03 ± 0.07	-0.03 ± 0.07	0.04 ± 0.05	–	–	0.04 ± 0.05
3-Actions	–	–	–	–	0.10 ± 0.05	0.12 ± 0.06	–	0.11 ± 0.03
Avg ± SEM	–	-0.08 ± 0.12	0.01 ± 0.07	-0.02 ± 0.04	0.07 ± 0.03	0.12 ± 0.06	–	0.08 ± 0.03

(g) ARC-RL

	Forgetting				Forward Transfer			
	7-Actions	5-Actions	3-Actions	Avg ± SEM	7-Actions	5-Actions	3-Actions	Avg ± SEM
7-Actions	–	-0.01 ± 0.02	-0.01 ± 0.01	-0.01 ± 0.01	–	–	–	–
5-Actions	–	–	0.15 ± 0.08	0.15 ± 0.08	0.90 ± 0.02	–	–	0.90 ± 0.02
3-Actions	–	–	–	–	0.91 ± 0.02	-0.01 ± 0.02	–	0.45 ± 0.01
Avg ± SEM	–	-0.01 ± 0.02	0.07 ± 0.04	0.04 ± 0.03	0.90 ± 0.01	-0.01 ± 0.02	–	0.60 ± 0.01



(a) Task 1: Five actions (b) Task 2: Three actions (c) Task 3: Seven actions

Figure 11: Performance of seven methods on three MiniGrid tasks in the **contraction & expansion** situation.

Table 9: Continual learning metrics of seven methods and two variants of ARC-RL across three MiniGrid tasks in combined situations of **expansion and contraction**. Continual return and forward transfer are abbreviated as "Return" and "Transfer", respectively. The top three results are highlighted in green, and the depth of the color indicates the ranking.

Methods	Expansion & Contraction			Contraction & Expansion		
	Return↑	Forgetting↓	Transfer↑	Return↑	Forgetting↓	Transfer↑
IND	0.78 ± 0.03	0.12 ± 0.02	–	0.79 ± 0.03	0.10 ± 0.02	–
FT	0.87 ± 0.03	0.04 ± 0.02	0.50 ± 0.02	0.88 ± 0.03	0.02 ± 0.01	0.39 ± 0.05
EWC	0.83 ± 0.03	0.01 ± 0.02	0.44 ± 0.04	0.40 ± 0.11	0.20 ± 0.05	0.30 ± 0.06
online-EWC	0.88 ± 0.02	-0.02 ± 0.02	0.45 ± 0.04	0.86 ± 0.03	0.04 ± 0.02	0.40 ± 0.05
Mask	0.68 ± 0.06	0.04 ± 0.04	0.01 ± 0.02	0.64 ± 0.05	0.03 ± 0.02	0.05 ± 0.03
CLEAR	0.51 ± 0.11	0.35 ± 0.06	0.54 ± 0.02	0.07 ± 0.02	0.39 ± 0.02	0.21 ± 0.03
ARC-RL	0.91 ± 0.01	-0.03 ± 0.01	0.55 ± 0.02	0.90 ± 0.03	0.03 ± 0.02	0.42 ± 0.03

task and contracts to five after the second task. Similarly, "contraction & expansion" represents the situation where the size of action space contracts to three after the first task and expands to seven after the second task. ARC-RL achieves near-best performance in terms of continual return, forgetting, and forward transfer in both situations. These results are consistent with the previous experiments, further demonstrating the advantages of ARC-RL in handling more complex situations.

Table 10: Continual learning metrics of seven methods in a longer sequence (five MiniGrid tasks). The top three results are highlighted in green, and the depth of the color indicates the ranking.

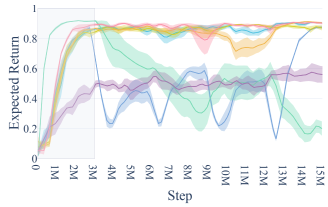
Methods	Metrics		
	Return↑	Forgetting↓	Transfer↑
IND	0.77 ± 0.06	0.08 ± 0.02	–
FT	0.76 ± 0.10	0.09 ± 0.04	0.29 ± 0.01
EWC	0.87 ± 0.02	0.00 ± 0.00	0.32 ± 0.01
online-EWC	0.74 ± 0.11	0.08 ± 0.05	0.16 ± 0.01
Mask	0.67 ± 0.07	0.05 ± 0.03	0.04 ± 0.02
CLEAR	0.14 ± 0.04	0.34 ± 0.01	0.29 ± 0.02
ARC-RL	0.89 ± 0.02	-0.01 ± 0.01	0.34 ± 0.01

C.3 SACLING TO LONGER SEQUENCE

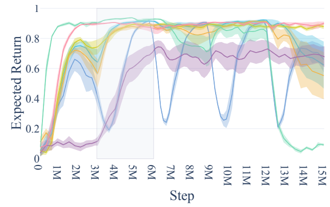
We also evaluate the CRL methods' performance over a longer sequence of SE-CRL. This sequence consists of five MiniGrid tasks, where the action space is expanding and then contracting. Each task is trained for a total of 15M environment steps, and results are reported over five runs. As shown in Figure 12 and Table 10, most methods' performance remains consistent with previous experiments, except for EWC, which performs better in this longer sequence. This improvement may be due to the repeated tasks in the sequence, benefiting EWC's regularization. ARC-RL achieves the best performance in terms of continual return, forgetting, and forward transfer, demonstrating its ability to handle combined situations of action space changes over longer task sequences.

C.4 HYPERPARAMETER SENSITIVITY ABLATIONNALYSIS

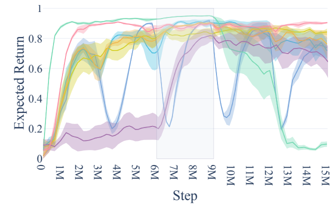
Figure 13 presents a hyperparameter sensitivity analysis of the action representation size and the regularization coefficient (λ) across three MiniGrid tasks. For comparison, the performance of FT (baseline) is also shown. In the expansion situation, both hyperparameters have minimal impact on the performance of ARC-RL. In the contraction situation, both hyperparameters slightly affect results. Forward transfer is positively correlated with the action representation size, but a smaller action representation size (128) may lead to a better continual return. A very small regularization coefficient (1000) can cause catastrophic forgetting and decreased continual return, further indicating the importance of regularization in the contraction situation. Overall, the contraction situation is more sensitive to hyperparameters than the expansion situation, but ARC-RL's performance remains relatively robust in both.



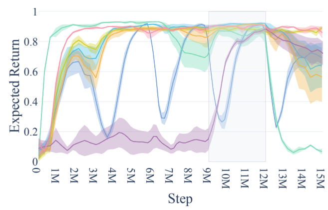
(a) Task 1: Seven actions



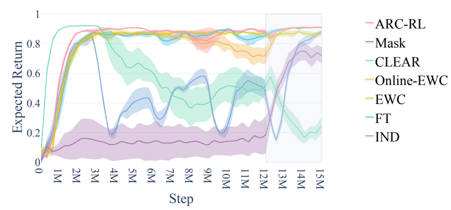
(b) Task 2: Five actions



(c) Task 3: Seven actions

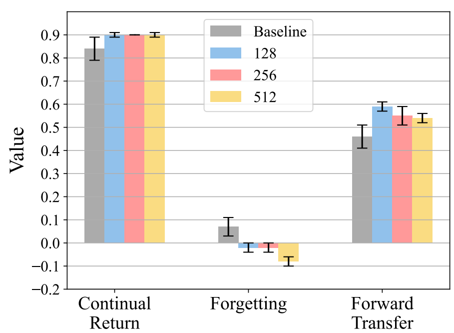


(d) Task 4: Five actions

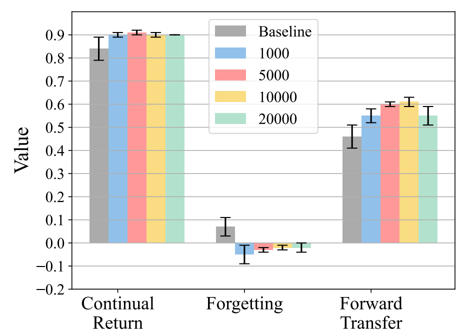


(e) Task 5: Three actions

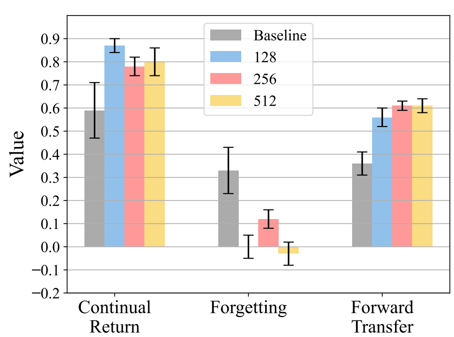
Figure 12: Performance of seven methods in a longer sequence (five MiniGrid tasks).



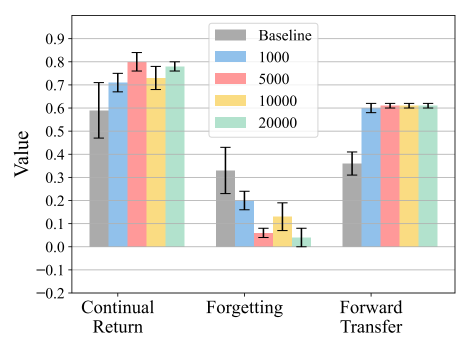
(a) Action representation size



(b) Regularization coefficient



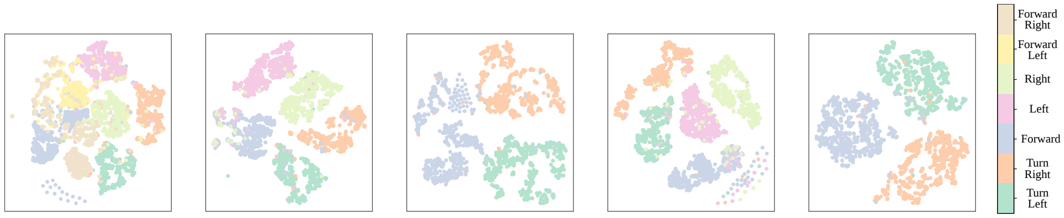
(c) Action representation size



(d) Regularization coefficient

Figure 13: Hyperparameter sensitivity analysis for ARC-RL across three MiniGrid tasks in the situations of expansion (**above**) and contraction (**bottom**). We examine the impact of the action representation size and the regularization coefficient (λ) in ARC-RL. Other hyperparameters are kept consistent with the competitive experiments, except that each experiment is repeated only five times. Error bars represent the standard error.

C.5 Visualization

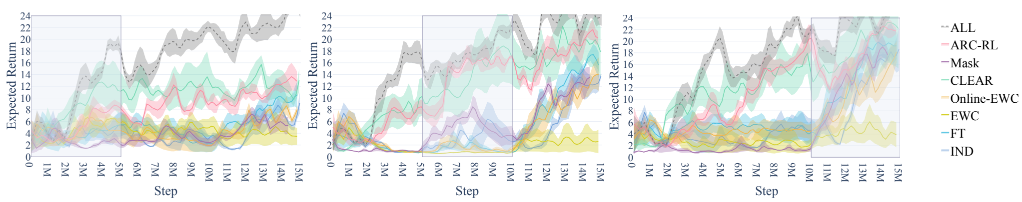


(a) Task 1 (b) Task 2 (c) Task 3 (d) Task 2 (e) Task 3

Figure 14: 2D t-SNE visualizations of learned action representations on MiniGrid tasks, colored by actual actions. The number of points for each action is 1000. **(a)–(c)**: Fine-tuning with regularization. **(d)–(e)**: Learning independently.

To observe how action representation space learned by ARC-RL changes with action spaces, we adopt t-SNE (Maaten & Hinton, 2008) to visualize the learned action representations on MiniGrid tasks in a 2D plane. Figure 14 shows that ARC-RL constructs a smooth representation space, where points with similar influence are clustered together. For instance, the points of the action "turn left" and "turn right" are close to each other. Although very similar actions are not well distinguished in the figure, this reflects their substitutability. Through regularized constraints, as the action space changes, removed actions are replaced by existing actions while maintaining their relative positions in the previous action space. In contrast, independently learned representations are redistributed, failing to preserve previous action relationships. This indicates that regularized fine-tuning in ARC-RL can maintain knowledge of the previous action space, benefiting continual learning.

C.6 Expansion Situation on Bigfish



(a) Task 1: Three actions (b) Task 2: Five actions (c) Task 3: Nine actions

Figure 15: Performance of eight methods on three Bigfish tasks in the **expansion** situation.

Figure 15 and Table 11 show the performance and metrics of ARC-RL and other methods in the expansion situation across three Bigfish tasks. ARC-RL achieves the best performance in terms of continual return and forward transfer. In this situation, all methods perform well in terms of mitigating forgetting, which is consistent with the results on MiniGrid tasks. Similarly, the overall performance of CLEAR is better than other methods, but ARC-RL still outperforms it.

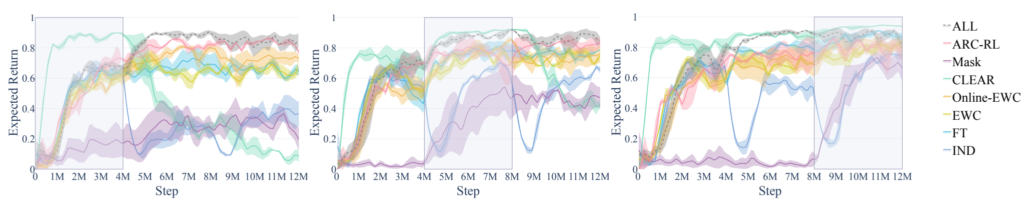
C.7 Other Environments

We also conducted experiments in other environments, including MiniGrid-Obstacles and Leaper. The former is a more challenging version of MiniGrid Empty, where the agent must navigate through obstacles to reach the goal. The latter is another environment from Procgen where the agent must jump over obstacles to reach the goal. The action space and task of MiniGrid-Obstacles are the same as MiniGrid Empty. For Leaper, the agent has five actions: jump up, jump down, jump left, jump right, and stay. Similarly, we design three tasks for Leaper with different action spaces: a two-action

Table 11: Continual learning metrics of eight methods in the **expansion** situation on three Procgen tasks. The average continual return of **ALL** is 24.77, which is not provided in the table. The top three results are highlighted in green, and the depth of the color indicates the ranking.

Methods	Metrics		
	Return↑	Forgetting↓	Transfer↑
IND	13.86 ± 2.41	-0.27 ± 0.08	–
FT	15.13 ± 0.10	-0.25 ± 0.04	0.00 ± 0.02
EWC	3.31 ± 1.88	-0.00 ± 0.11	-0.08 ± 0.12
online-EWC	13.43 ± 2.02	-0.31 ± 0.09	0.02 ± 0.05
Mask	12.18 ± 1.45	-0.19 ± 0.04	-0.02 ± 0.03
CLEAR	17.68 ± 4.79	-0.04 ± 0.03	0.18 ± 0.07
ARC-RL	18.27 ± 1.22	-0.05 ± 0.11	0.21 ± 0.05

task (jump up, stay), a three-action task (jump up, jump down, stay), and a five actions (jump up, jump down, jump left, jump right, stay). The action space changes in these tasks are not as smooth as in Bigfish tasks, making differences between compared methods not as significant. Figure 16 and Table 12 show the performance of ARC-RL and other methods in the expansion situation on three MiniGrid-Obstacles tasks. Each task is replicated with three random seeds. Figure 17 and Table 13 show the performance of ARC-RL and other methods on three Leaper tasks. Each task is replicated with three random seeds. The results show that ARC-RL also achieves the best performance in terms of continual return in both environments.



(a) Task 1: Three actions (b) Task 2: Five actions (c) Task 3: Seven actions

Figure 16: Performance of eight methods on three MiniGrid-Obstacles tasks in the **expansion** situation.

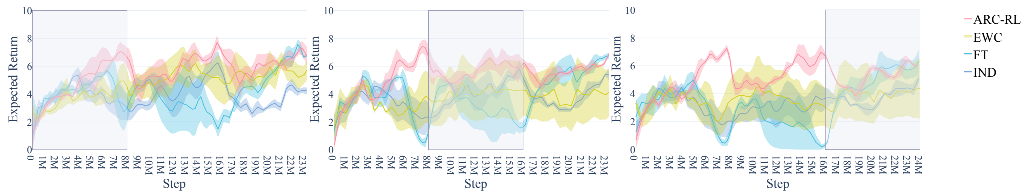
Table 12: Continual learning metrics of eight methods in the **expansion** situation on three MiniGrid-Obstacles tasks. The average continual return of **ALL** is 0.86, which is not provided in the table. The top three results are highlighted in green, and the depth of the color indicates the ranking.

Methods	Metrics		
	Return↑	Forgetting↓	Transfer↑
IND	0.13 ± 0.02	0.25 ± 0.06	–
FT	0.75 ± 0.05	0.01 ± 0.04	0.45 ± 0.02
EWC	0.68 ± 0.03	0.06 ± 0.11	0.42 ± 0.04
online-EWC	0.74 ± 0.05	-0.02 ± 0.10	0.38 ± 0.04
Mask	0.40 ± 0.07	-0.07 ± 0.10	-0.05 ± 0.05
CLEAR	0.68 ± 0.06	0.27 ± 0.06	0.56 ± 0.03
ARC-RL	0.80 ± 0.04	-0.06 ± 0.01	0.47 ± 0.07

D More Discussion

E Connection with Neuroscience

The field of neuroscience offers valuable insights into the mechanisms underlying motor control and learning, which can inform the development of artificial intelligence systems, particularly in the context of CRL (Kaplanis et al., 2019; Gazzaniga et al., 2019; Kudithipudi et al., 2022).



(a) Task 1: Two actions (b) Task 2: Three actions (c) Task 3: Five actions

Figure 17: Performance of four methods on three Leaper tasks in the **expansion** situation.

Table 13: Continual learning metrics of four methods in the **expansion** situation on three Leaper tasks. The top results are highlighted in green.

Methods	Metrics		
	Return↑	Forgetting↓	Transfer↑
IND	5.03 ± 0.49	-0.05 ± 0.05	–
FT	6.31 ± 0.62	-0.26 ± 0.01	-0.11 ± 0.04
EWC	4.72 ± 1.80	-0.10 ± 0.04	0.11 ± 0.11
ARC-RL	6.69 ± 0.48	-0.03 ± 0.08	0.42 ± 0.05

In the human brain, motor control is distributed across several anatomical structures that operate hierarchically (D'Mello et al., 2020; Friedman & Robbins, 2022). At the highest levels, planning is concerned with how an action achieves an objective, while lower levels translate goals into specific movements. This hierarchical organization allows for flexible and adaptive behavior, as higher-level goals can be achieved through various lower-level actions depending on the context. Similarly, in ARC-RL, the agent's policy can be seen as operating at a high level, focusing on achieving task objectives, while the action representation space operates at a lower level, translating these objectives into specific actions. By decoupling the policy from the specific action space, ARC-RL leverages a hierarchical approach that mirrors the brain's strategy for motor control. This allows the agent to adapt to changes in the action space without needing to relearn the entire policy.

Neurophysiological studies have shown that the activity of neurons in the motor cortex is often correlated with movement direction rather than specific muscle activations (Georgopoulos & Pellizzer, 1995; Kakei et al., 1999). Neurons exhibit directional tuning, and their collective activity can be represented as a population vector that predicts movement direction. This concept of population coding suggests that the brain represents actions in a high-dimensional space, allowing for generalization across different contexts. In ARC-RL, the action representation space serves a similar function. By encoding actions in a high-dimensional space, the agent can generalize its policy across different action spaces. The encoder-decoder architecture in ARC-RL can be likened to the neural mechanisms that map cortical activity to specific movements. When the action space changes, the update of the ecoder and the decoder, is akin to how the brain might update its motor representations in response to changes in the body or environment.

Recent research in motor neurophysiology has highlighted the dynamic nature of neural representations. Neurons do not have fixed roles but instead can represent different features depending on the context and time (Churchland et al., 2012; Gallego et al., 2020). This flexibility allows the motor system to adapt to a wide range of tasks and environments, providing maximum behavioral flexibility. ARC-RL incorporates this idea by allowing the action representation space to be dynamically updated. This dynamic updating process is analogous to how the brain adjusts its neural representations to maintain consistent behavior despite changes in the body or environment.

By drawing inspiration from neuroscience, ARC-RL achieve policy generalization and adaptability in the face of changing action spaces. This connection between neuroscience and artificial intelligence not only enhances our understanding of both fields but also provides feasible ideas for more sophisticated and adaptable AI systems.

E.1 Extending to Other Situations

Our framework can be extended to other situations as it is not specifically designed for expansion and contraction situations. Both the partial change situation and the complete change situation can be viewed as simultaneous occurrences of expansion and contraction. In the partial change situation, some actions from the previous action space remain, while in the complete change situation, all previous actions are removed.

In the complete change situation, incorporating experience replay is a straightforward improvement of our framework. However, this approach incurs additional storage and computational costs. Thus, the trade-off between performance and efficiency is necessary. In addition, the method of generating pseudo samples (Shin et al., 2017; Yue et al., 2023) via representation space may be more suitable for practical privacy-preserving scenarios, where the agent cannot access the previously collected data.

For the partial change situation, the challenge lies in identifying the differences between action spaces. Our framework can be improved by integrating mechanisms to detect removed and newly added actions, similar to novelty detection and class-incremental learning in the open-world setting (Masana et al., 2022; Sahisnu Mazumder, 2024; Li et al., 2024a). While clustering or classifying action representations can achieve this, it imposes higher demands on the self-supervised learning method. The action representations need to be well-separated in the representation space. Therefore, leveraging more information beyond state changes to learn action representations could be a valuable extension of our framework.

E.2 Limitations and Future Work

Despite the promising advancements, we acknowledge the current limitations, primarily the scalability concerns when dealing with large action spaces in complex environments. Future avenues of research should focus on enhancing adaptability by incorporating existing exploration methods and developing more effective action representation learning algorithms. In addition, more complex changes in the action space, such as the changes of continuous action space, require further investigation. This require more sophisticated representation learning methods to capture the action representations. The exploration policy in ARC-RL is also not suitable for continuous action spaces, and future work should explore more advanced exploration strategies for such scenarios. Moreover, we plan to extend our work to a broader range of CRL, including changes in the agent's capabilities and the external environment. This may involve utilizing meta-learning strategies to improve generalization across a broader spectrum of capabilities evolutions and incorporating advanced transfer learning mechanisms to seamlessly integrate knowledge from a wider range of environments and varying agent capabilities.