KNOWLEDGE-GROUNDED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Receiving knowledge, abiding by laws, and being aware of regulations are common behaviors in human society. Bearing in mind that reinforcement learning (RL) algorithms benefit from mimicking humanity, in this work, we propose that an RL agent can act on external guidance in both its learning process and model deployment, making the agent more socially acceptable. We introduce the concept, Knowledge-Grounded RL (KGRL), with a formal definition that an agent learns to follow external guidelines and develop its own policy. Moving towards the goal of KGRL, we propose a novel actor model with an embedding-based attention mechanism that can attend to either a learnable internal policy or external knowledge. The proposed method is orthogonal to training algorithms, and the external knowledge can be flexibly recomposed, rearranged, and reused in both training and inference stages. Through experiments on tasks with discrete and continuous action space, our KGRL agent is shown to be more sample efficient and generalizable, and it has flexibly rearrangeable knowledge embeddings and interpretable behaviors.

1 INTRODUCTION

Incorporating external guidance into learning is a commonly seen behavior among humans. We can speed up our learning process by referring to useful suggestions. At the same time, we beware of external regulations for safe and ethical reasons. On top of following external guidance, we humans can learn our own strategies to complete a task. We can also arbitrarily recompose, rearrange, and reuse those strategies and external guidelines to solve a new task and adapt to environmental changes. Imitating human behaviors has been shown to benefit reinforcement learning (RL) (Billard et al., 2016; Sutton and Barto, 2018; Zhang et al., 2019). However, how an RL agent can achieve the above capabilities remains challenging.

Different approaches have been proposed to develop some of these abilities. One branch of previous work in RL has studied how an agent can learn from demonstrations provided externally as examples of completing a task (Ross et al., 2011; Rajeswaran et al., 2017; Duan et al., 2017; Nair et al., 2018; Goecks et al., 2019; Ding et al., 2019). Another branch of previous research has investigated how an agent can learn reusable policies. This branch of research includes (1) transferring knowledge among tasks with similar difficulty (Parisotto et al., 2015; Yin and Pan, 2017; Gupta et al., 2018a; Liu et al., 2019; Tao et al., 2021) and (2) learning multiple reusable skills to solve a complex task in a divide-and-conquer manner (Bacon et al., 2017; Frans et al., 2017; Nachum et al., 2018a; Eysenbach et al., 2018; Kim et al., 2021; Tseng et al., 2021). These methods allow an agent to learn a new task with fewer training samples. However, demonstrations are too task-specific to be reused in a new task, and incorporating external guidelines from different sources into existing knowledge-reuse frameworks is not straightforward. Moreover, current learning-from-demonstration and knowledge-reuse approaches lack the flexibility to rearrange and recompose different demonstrations or knowledge, so they cannot dynamically adapt to environmental changes.

In this work, we introduce Knowledge Grounded Reinforcement Learning (KGRL), a novel problem with the following goal: *An RL agent can learn its own policy (knowledge) while referring to external knowledge. Meanwhile, all knowledge can be arbitrarily recomposed, rearranged, and reused anytime in the learning and inference stages.* A KGRL problem simultaneously considers the following three questions: (1) How can an agent follow a set of external knowledge from different sources? (2) How can an agent efficiently learn new knowledge by referring to external ones? (3) What is a proper

representation of knowledge such that an agent can dynamically recompose, rearrange, and reuse knowledge in both the learning process and model deployment?

To address the challenges in KGRL, we propose a simple yet effective actor model with an embedding-based attention mechanism. This model is orthogonal to training algorithms and easy to implement. In this model, each external-knowledge policy is paired with one learnable embedding. An internal-knowledge policy, which the agent learns by itself, is also paired with one learnable embedding. Then another learnable query embedding performs an attention mechanism by attending to each knowledge embedding and then decides which knowledge to follow. With this attention mechanism, the agent can learn new skills by following a set of external guidance. In addition, the knowledge and query embeddings design disentangles each knowledge from the selection mechanism, so a set of knowledge can be dynamically recomposed or rearranged. Finally, all knowledge policies are encoded into a joint embedding space, so they can be reused once their embeddings are appropriately learned.

We evaluate our proposed KGRL method for grid navigation and robotic manipulation tasks. The results demonstrate that our method answers all the questions considered in KGRL. In analyses, we show that the proposed approach achieves sample efficiency, generalizability, compositionality, and incrementality, which are the four essential components of efficient learning (Kaelbling, 2020). At the same time, our method also enables behavior interpretability because of the embedding-based attention mechanism.

Our contributions are:

- We introduce KGRL, a novel RL problem addressing how an agent can repeatedly refer to an arbitrary set of external knowledge for learning new skills.
- We propose a novel actor model that achieves the goals of KGRL and is orthogonal to training algorithms.
- We demonstrate in the experiments that our KGRL method satisfies the four properties of efficient learning (Kaelbling, 2020) and learns interpretable strategies.

2 Related Work

Several lines of research in RL have studied how an agent can incorporate external demonstrations/regulations into learning or learn reusable policies. Here we summarize the differences between their research goals and the ones in the KGRL formulation.

Learning from demonstrations (LfD) and imitation learning (IL). Providing expert demonstrations to an RL agent is a popular way to incorporate external guidance into learning. Demonstrations are (sub-)optimal examples of completing a task and represented as state-action pairs. Previous work has leveraged demonstrations by introducing them into the policy-update steps of RL (Hester et al., 2017; Rajeswaran et al., 2017; Vecerik et al., 2017; Nair et al., 2018; Pfeiffer et al., 2018; Goecks et al., 2019) or performing IL (Abbeel and Ng, 2004; Ross et al., 2011; Ho and Ermon, 2016; Duan et al., 2017; Peng et al., 2018a;b; Ding et al., 2019). LfD and IL require sufficient number of optimal demonstrations, which are difficult to collect, to achieve sample-efficient learning. Moreover, those demonstrations are task-dependent and can hardly be reused in other tasks. Compared to LfD and IL, KGRL can achieve efficient learning by referring to highly imperfect knowledge. Also, KGRL can incorporate not only task-related knowledge but also knowledge of commonsense, principles, or regulations to move toward safe and ethical learning. Finally, all knowledge, including the one learned by the agent itself, can be reused in new tasks.

Safe RL. Other than demonstrations, regulations are also essential guidance that can help an agent achieve effective learning. Safe RL, which incorporates safety regulations (constraints) into learning and inference stages, has gained more attention over the past decade. Current research in safe RL mainly includes safety in learning through (1) jointly optimizing the expected return and safe-related cost (Ammar et al., 2015; Achiam et al., 2017; Chow et al., 2018; Stooke et al., 2020; Ding et al., 2021) and (2) restricting exploration in a safe space (Garcia and Fernández, 2012; Berkenkamp et al., 2017; Alshiekh et al., 2018; Cheng et al., 2019; Thananjeyan et al., 2021; Liu et al., 2022). The constraints in safe RL can be softly or strictly imposed, where the former aims to decrease the

number of safety violations, and the latter prohibits any of them in both learning and inference stages. Compared to KGRL, safe RL usually does not consider reusing the safety constraints in a new task or dynamically adjusting regulations according to the environmental changes.

Knowledge transfer and meta-RL. How an agent can transfer previously learned knowledge to a new task is a well-recognized challenge. Several studies have focused on transferring the knowledge learned in a single task to a multi-task policy (Parisotto et al., 2015; Rusu et al., 2015; Yin and Pan, 2017; Xu et al., 2020; Tao et al., 2021). On the other hand, a line of research in meta-RL investigates how to quickly adapt the knowledge learned in multiple tasks to a new one (Finn et al., 2017; Gupta et al., 2018a;b; Nagabandi et al., 2018; Liu et al., 2019). These two research directions do not enable incorporating external guidance from different sources into learning, so their transferred knowledge is limited to the one learned among similar environments (Glatt et al., 2016). In addition, they require extensive amount of training data before a set of knowledge can be reused in a new task. Compared to previous work in knowledge transfer and meta-RL, KGRL allows an agent to follow and reuse an arbitrary set of knowledge obtained from different sources in a new task, and these sources can be very different from the new task. Moreover, our KGRL approach does not require a separate process, such as policy distillation (Rusu et al., 2015), or numerous training samples collected from multiple environments to learn transferable knowledge.

Hierarchical RL (HRL). Other than transferring knowledge among tasks with similar difficulty, another popular line of research, HRL, aims to decompose a complex task into a hierarchy of subtasks and learn a reusable skill for each subtask. Previous research in HRL has focused on learning a hierarchy of a complex task (Dayan and Hinton, 1992; Sutton et al., 1999; Kulkarni et al., 2016; Levy et al., 2017; Nachum et al., 2018b; Jiang et al., 2019), discovering options/subtasks/subgoals (Stolle and Precup, 2002; Bacon et al., 2017; Vezhnevets et al., 2017; Nachum et al., 2018a; Riemer et al., 2018; Eysenbach et al., 2018; Khetarpal and Precup, 2019; Zhang and Whiteson, 2019; Kim et al., 2021), and transferring learned skills among different tasks (Konidaris and Barto, 2006; Frans et al., 2017; Tessler et al., 2017; Peng et al., 2019; Qureshi et al., 2019; Tseng et al., 2021). Compared to the skills in HRL, knowledge in KGRL can be partially contained in, be more complex than, or even has little to do with the task. That is, it is not limited to a skill that solves a subtask. This property enables flexibility of providing knowledge and improves exploration in an unseen environment. In addition, a skill in HRL is usually assigned with a fixed index or a task-dependant subgoal, making an agent difficult to dynamically recompose or rearrange skills in learning or inference stages. On the other hand, our KGRL method disentangles knowledge representation from the high-level selection mechanism of a task, making an agent able to dynamically adapt to environmental changes anytime.

Program-guided RL. Similar to HRL, program-guided RL also decomposes a complex task into subtasks and learns subpolicies for them. However, instead of learning a hierarchy of the task, program-guided RL guides an agent to complete a task with a program, which explicitly specifies the flow of executing the subtasks given the environmental conditions (Sun et al., 2019; Brooks et al., 2021; Trivedi et al., 2021; Yang et al., 2021; Zhao et al., 2021). This guidance requires a structured and unambiguous hierarchy of a task and is more task-dependent compared to knowledge in KGRL. Moreover, to reuse a previously learned subpolicy, program-guided RL needs to first generate a new program with a new control flow, hence lacking the flexibility to dynamically recompose and rearrange subpolicies.

3 KGRL Problem Formulation

In this section, we formally define the problem formulation of *Knowledge-Grounded Reinforcement Learning (KGRL)*. We consider the family of discrete-time finite-horizon Markov Decision Processes (MDPs) with knowledge guidance for an environment. Such an MDP is formulated as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{G}, \mathcal{R}, \rho, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the transition probability distribution, $\mathcal{G}$ is the set of *high-level knowledge*, $\mathcal{R}$ is the reward function, ρ is the initial state distribution, and γ is the discount factor.

Each knowledge in the high-level knowledge set, $g \in \mathcal{G}$, can be any commonsense, suggestion, guideline, principle, specification, rule, restriction, or previously learned knowledge described in

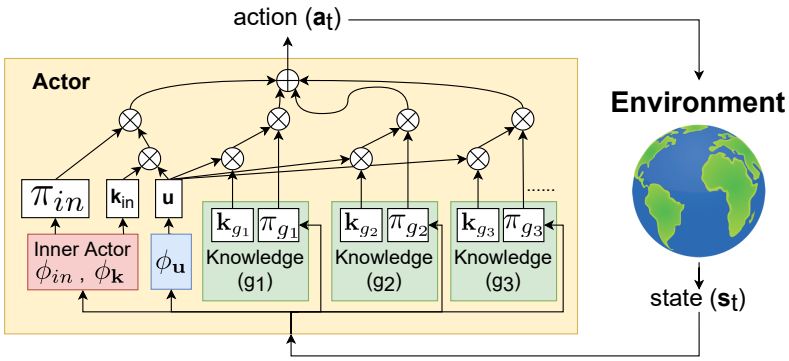


Figure 1: The proposed model architecture of a KGRL agent.

formal or informal language. Note that each $g \in \mathcal{G}$ will be transformed to a low-level knowledge mapping $\pi_g : \mathcal{S} \to \mathcal{A}$, which describes its low-level execution strategy.

The goal of a KGRL agent is to maximize the expected rewards and develop the following attributes: (1) The agent learns its own knowledge by referring to external one. (2) The agent can recompose/rearrange/reuse knowledge in training and inference stages as well as new environments. (3) The agent's behaviors are interpretable in terms of which knowledge it refers to at each time step.

4 Proposed Approach

We propose a novel actor model for KGRL with the concept inspired by the self-attention mechanism (Vaswani et al., 2017). The actor model includes an inner actor and a set of external knowledge mappings, $\{\pi_{g_1}, \pi_{g_2}, \ldots, \pi_{g_n}\}$. The actor architecture is illustrated in Figure 1. The idea behind this design is that the actor will decide to follow an external guidance or its own policy given a state $\mathbf{s}_t \in \mathcal{S}$ at the t-th time step. This dynamic decision making should eventually lead the agent to receive higher accumulated rewards based on the reward function $\mathcal{R}$. Moreover, the external knowledge set can be recomposed/rearranged/reused anytime during training and testing phases. In the following sections, we introduce the detailed design of each component in the proposed actor model.

4.1 Actor Model Architecture

External Knowledge. An important property of a KGRL policy is the flexibility of knowledge recomposition/rearrangement/reusability during training and testing phases. To obtain this flexibility, we propose a uniform representation for each knowledge. We learn a *key* (embedding) for each external knowledge mapping, which is a d_k-dimensional vector denoted as $\mathbf{k}_{g_j} \in \mathbb{R}^{d_k}$, for $j = 1, 2, \ldots, n$. Note that the knowledge key $\mathbf{k}_{g_j}$ is for the *entire* knowledge mapping π_{g_j} instead of each state-action pair suggested by this mapping. Since each knowledge is encoded as an *independent* key, which lies in a joint embedding space, one can change the number of knowledge by adding/removing keys to/from the actor model any time during training and testing. Meanwhile, one can also combine knowledge keys from different tasks in a new learning/evaluation process.

Inner Actor. An inner actor contains two function approximators ϕ_{in} and $\phi_{\mathbf{k}}$. Given a state $\mathbf{s}_t$, the function approximator ϕ_{in} predicts an inner policy, $\pi_{in} = \phi_{in}(\mathbf{s}_t) : \mathcal{S} \to \mathcal{A}$, and $\phi_{\mathbf{k}}$ generates an *inner key*, $\mathbf{k}_{in} = \phi_{\mathbf{k}}(\mathbf{s}_t) \in \mathbb{R}^{d_k}$. The inner policy serves the same purpose as a policy in regular RL, which represents the strategy that an agent learns by itself through interactions with the environment. Similar to an external knowledge mapping π_{g_i}, the inner policy π_{in} can be viewed as the *internal knowledge mapping* with a paired learnable inner key $\mathbf{k}_{in}$. Note that unlike a knowledge key, the inner key varies according to a given state, because the inner strategy will change over time.

Query. To decide which inner/external knowledge to refer to, we add another function approximator $\phi_{\mathbf{u}}$ that generates a *query* vector, denoted as $\mathbf{u} = \phi_{\mathbf{u}}(\mathbf{s}_t) \in \mathbf{R}^{d_k}$, given a state $\mathbf{s}_t$. The query together with the keys will be used to calculate the weights for each knowledge mapping. Note that the proposed model predicts the query and keys separately instead of directly predicting the weight for

Algorithm 1: Knowledge-Guided Reinforcement Learning (KGRL) with Our Actor Model

Input: environment $\mathcal{E}$ with MDP $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{G}, \mathcal{R}, \rho, \gamma)$
Input: external knowledge mappings $\pi_{g_1}, \pi_{g_2}, ..., \pi_{g_n}$
initialize $\mathbf{k}_{g_j}$ for $j = 1, 2, ..., n$
initialize $\phi_{in}, \phi_{\mathbf{k}}, \phi_u$
for *each time step* **do**
  Generate $\mathbf{k}_{g_1}, \ldots, \mathbf{k}_{g_n}, \mathbf{k}_{in}, \pi_{in}, \mathbf{u}$
  Compute $w_{in}, w_{g_1}, \ldots, w_{g_n}$ according to equation (1)
  Sample an action $\mathbf{a}_t \sim \pi(\cdot|\mathbf{s}_t)$ according to equations (2)
  Observe a new state and reward $\mathbf{s}_{t+1}, r_t$ from $\mathcal{E}$
  Update $\phi_{in}, \phi_k, \phi_u, \mathbf{k}_{g_1}, \ldots, \mathbf{k}_{g_n}$ by any RL algorithm
return $\phi_{in}, \phi_{\mathbf{k}}, \phi_u, \mathbf{k}_{g_1}, \ldots, \mathbf{k}_{g_n}$

each guidance. In this way, we disentangle the decision component (the query) and the knowledge representations (the keys). The following section describes how we compute the weights.

4.2 Action Prediction

Attentions. We compute the attention of the query $\mathbf{u}$ over the inner keys and external knowledge keys by their dot products. The results become the weights for the inner and external guidance:

$$\begin{aligned} w_{in} &= \mathbf{u} \cdot \mathbf{k}_{in} \,, \\ w_{g_j} &= \mathbf{u} \cdot \mathbf{k}_{g_j} \,. \end{aligned} \tag{1}$$

If w_{in} is larger, then the agent is more likely to adopt the inner policy. On the other hand, if w_{g_j} is larger, then the agent is more likely to follow the guidance from the j-th external knowledge.

Final Policy. A straightforward way to describe the final policy given the weights w_{in} and w_{g_j} and their corresponding knowledge mappings π_{in} and π_{g_j}, for $j = 1, 2, \ldots, n$, is

$$\mathbf{a}_t \sim \pi(\cdot|\mathbf{s}_t), \tag{2}$$

$$\pi(\cdot|\mathbf{s}_t) = w_{in}\pi_{in}(\cdot|\mathbf{s}_t) + \sum_{j=1}^{n} w_{g_j}\pi_{g_j}(\cdot|\mathbf{s}_t), \tag{3}$$

which is a weighted sum of each strategy. For a stochastic policy in a continuous action space, equation (3) becomes the probabilistic density function of a Gaussian Mixture Model (GMM). To sample an action from a GMM that makes the entire actor model differentiable, we first sample an element from the set $\{in, g_1, g_2, \ldots, g_n\}$ according to the weights $\{w_{in}, w_{g_1}, w_{g_2}, \ldots, w_{g_n}\}$ using Gumbel softmax (Jang et al., 2016):

$$e \sim \texttt{gumbel_softmax}(w_{in}, w_{g_1}, w_{g_2}, ..., w_{g_n})\,. \tag{4}$$

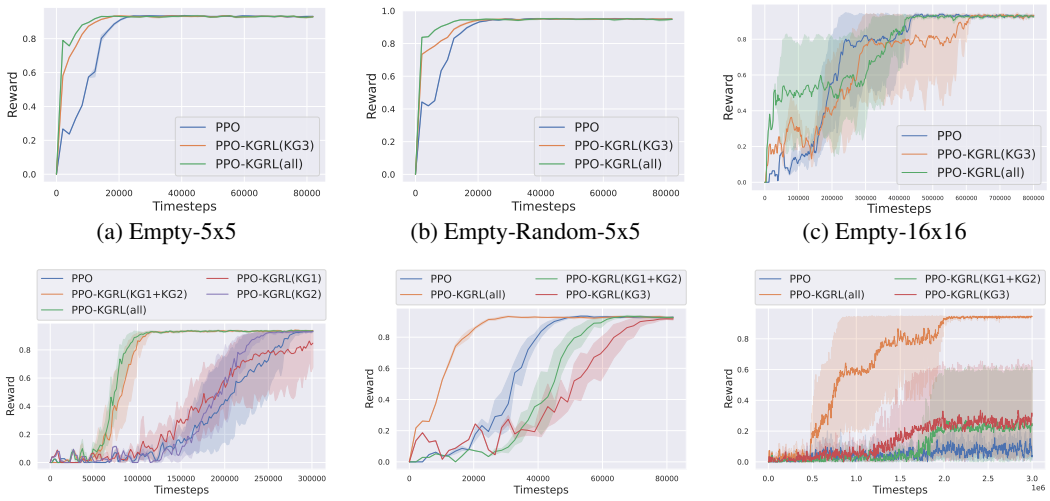
Then given a state $\mathbf{s}_t$, an action is sampled from the knowledge mapping $\pi_e(\cdot|\mathbf{s})$ using the reparameterization trick, where ϕ_e can be one of $\{\phi_{in}, \phi_{g_1}, \ldots, \phi_{g_n}\}$ according to the sampled e.

4.3 Learning Algorithms

The proposed actor model can be easily learned with any existing RL algorithm, and the implementation is simple. Algorithm 1 describes our proposed KGRL algorithm.

5 Experiments

We validate the proposed KGRL actor model on two sets of widely used environments with different properties: MiniGrid (Chevalier-Boisvert et al., 2018) and OpenAI-Robotics (Plappert et al., 2018). Through experiments on these environments, we answer the following four questions: (1) Can our KGRL agent learn effective new knowledge by referring to an arbitrary set of external guidance? (2) Can the knowledge policies and their learned embeddings be recomposed, rearranged, and reused to solve a new task? (3) Is our KGRL method sample efficient and generalizable? (4) Does our KGRL agent learn interpretable behaviors? The implementation details can be found in Appendix.



(a) Empty-5x5 (b) Empty-Random-5x5 (c) Empty-16x16

(d) Unlock (e) DoorKey-5x5 (f) DoorKey-8x8

Figure 2: The learning curves of RL and KGRL agents in MiniGrid environments. Our KGRL method achieves better sample efficiency in most of the environments. Also, adding redundant knowledge in the proposed model does not affect the performance.

5.1 MINIGRID

Environments. We evaluate the proposed method in the following MiniGrid environments with discrete action space and sparse rewards: Empty-5x5, Empty-Random-5x5, Empty-16x16, Unlock, DoorKey-5x5, and DoorKey-8x8. Among them, DoorKey are the environments that an agent needs to pick up a key, open a door, and then reach a goal, which can use the skills learned in Unlock (pick up a key and open a door) and Empty (reach a goal). We choose these environments to study if our method can reuse previously learned knowledge. The observation space of these environments is the directed first person view with a range of 5x5 grid, and the action space has seven discrete actions.

Setup. We compare Proximal Policy Optimization algorithm (Schulman et al., 2017) under RL (PPO) and KGRL (PPO-KGRL) formulations. In PPO-KGRL, we use the inner actor model same as the actor model of PPO and add the learnable inner key, knowledge keys, and query. We define three simple knowledge policies for the MiniGrid environments. **(KG1)** pick up a key: if there is a key in the observation, "go in the direction" of the key; if the key is just in front of the agent, pick up the key. **(KG2)** open a door: if there is a door in the observation, "go in the direction" of the door; if the door is just in front of the agent, open the door. **(KG3)** reach a goal: if there is a goal in the observation, "go in the direction" of the goal.

Results. Figure 2 shows the training results of PPO and PPO-KGRL with different knowledge combinations. Compared to PPO, PPO-KGRL achieves significantly better sample efficiency in five of the six environments. Also in general, the improvement is larger when the task is more complex (Empty $<$ Unlock $<$ DoorKey). Note that PPO does not show any success in DoorKey-8x8 within three million steps. However, we found that in Empty-16x16, our KGRL agent learns more quickly at the beginning but slows down its progress over time. The reason might be that Empty-16x16 requires only exploration instead of other skills, so the agent cannot benefit too much by following external guidance. Moreover, in a small environment such as DoorKey-5x5, the advantages of providing external knowledge is difficult to demonstrate due to extra learnable parameters in our actor model.

We also study the effects of different knowledge combinations for PPO-KGRL. In Unlock, we found that KG2 is more helpful than KG1; in DoorKey-8x8, KG3 is more helpful than KG1+KG2. We conjecture that the knowledge which directly helps receive the rewards guides an agent more effectively in these environments. Also, as shown in Figure 2(a-d), adding redundant knowledge will not affect the performance. This observation demonstrates the benefit of disentangling the knowledge embeddings and query, thus increasing the flexibility of providing external guidance.

Policy Interpretation. Figure 3 shows that our KGRL method learns interpretable agent behaviors. The weight changes indicate that in a DoorKey-8x8 environment, the agent chooses to sequentially "pick up the key", "open the door", "search for the goal", and "reach the goal". Note that the attention to the inner policy does not disappear throughout the entire episode. This shows that the inner policy helps an agent explore the environment.

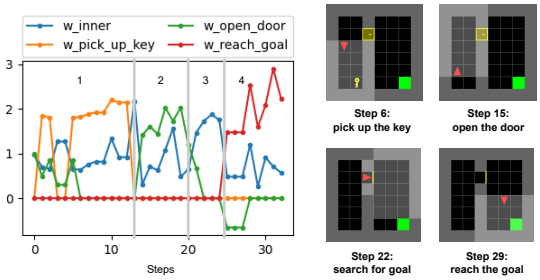


Figure 3: The interpretation of the agent's behaviors learned by our KGRL approach. The left image shows the variations of the (unnormalized) weights for each external and internal knowledge.

Analysis of Generalizability. To test the generalizability of PPO and PPO-KGRL, we train an agent in an empty environment with a fixed goal (Empty) and evaluate the agent in an empty environment with a randomized goal (Empty-Random). This experimental design verifies whether the agent memorizes the goal position or really understands how to find the goal. As shown in the left column of Table 1, the returns of PPO-KGRL achieve an average of 0.94 and a minimum of 0.77. This is much closer to the upper bound, which is the average return of a PPO-KGRL agent trained on the Empty-Random environment.

Init. Env.	Empty	DoorKey-5x5			DoorKey-8x8
Test Env.	Empty-Random	Empty	Empty-Random	Unlock	Unlock
PPO	0.92 (0.71)	0.55 (0.05)	0.79 (0.03)	0.43 (0)	0 (0)
PPO-KGRL	**0.94 (0.77)**	**0.88 (0.55)**	**0.90 (0.55)**	**0.54 (0)**	**0.85 (0)**
Upper Bound	0.95 (0.84)	0.93 (0.87)	0.95 (0.84)	0.94 (0.84)	0.94 (0.84)

Table 1: Zero-shot evaluations on three environments (Empty, Empty-Random, Unlock) of RL agents trained on an initial environment (Empty and DoorKey) using PPO or PPO-KGRL algorithms. The average and minimal returns are shown for each evaluation. Our KGRL method is more generalizable and effective in transferring previously learned knowledge.

Analysis of Reusing, Recomposing, and Rearranging Knowledge. For our KGRL method, we verify if the external and internal knowledge policies and their embeddings can be reused by the following two sets of experiments: (1) zero-shot transferring the knowledge policies and embeddings to a simpler task and (2) initializing the external guidance by those knowledge policies and embeddings when learning a more complex task. The second set of experiments also verifies if the knowledge policies and embeddings can be recomposed and rearranged.

In the first set of experiments, we zero-shot evaluate an agent trained in DoorKey environments on Empty, Empty-Random, and Unlock environments. The four columns on the right side of Table 1 show the average and minimal returns. We found that an agent trained with PPO-KGRL significantly outperforms the one trained with PPO in all environments. These results demonstrate that our KGRL policies have better reusability.

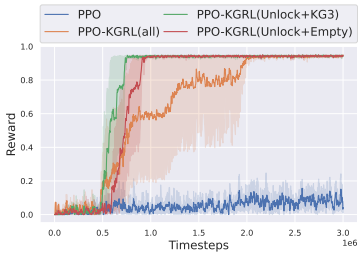
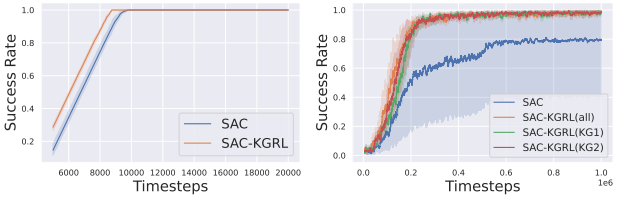


Figure 4: The knowledge reusability experiments on MiniGrid-DoorKey-8x8. Our KGRL method further improve sample efficiency by reusing previously learned knowledge.

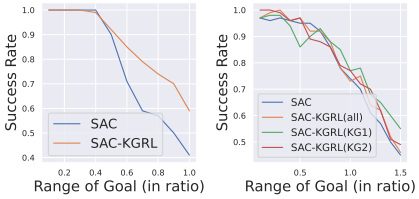
In the second set of experiments, we train three PPO-KGRL agents with external knowledge being:

1. all: KG1 + KG2 + KG3. This set of knowledge is the same one used in Figure 2.
2. Unlock (reused) + KG3: the inner knowledge policy and embeddings learned in Unlock + KG3. This set of knowledge recomposes and rearranges the first set of knowledge.



(a) Reach. (b) Pick-And-Place.

Figure 5: The learning curves of RL and KGRL agents in OpenAI robotic environments. Our KGRL method leads to better sample efficiency and stability in both environments.



(a) Reach. (b) Pick-And-Place.

Figure 6: Results of the policy trained on 0.1x the original goal range. Our KGRL method is less affected by the changes in the environment.

3. Unlock (reused) + Empty (reused): the inner knowledge policy and embeddings learned in Unlock + the ones learned in Empty-Random. This set of knowledge recomposes and rearranges the second set of knowledge.

Figure 4 shows the training results of PPO and PPO-KGRL with different sets of external knowledge. The two KGRL agents that reuse previously learned knowledge perform better than PPO and PPO-KGRL(all). Also surprisingly, we found that the number of samples required by PPO-KGRL that reuses the Unlock knowledge *plus* the one required by learning the Unlock knowledge is even less than the one required by PPO-KGRL(all), which learns with KG1+KG2+KG3. These results indicate that with the abilities to flexibly reuse, recompose, and rearrange knowledge, our KGRL method can further enhance the training efficiency.

5.2 Robotics

Environments. We also verify the proposed KGRL approach in the following OpenAI robotic environments with a continuous action space and sparse rewards: Reach and Pick-And-Place. Both environments contain a task of reaching a randomly sampled goal. Similar to the MiniGrid environments, these two tasks can demonstrate the ability of our KGRL method reusing, recomposing, and rearranging previously learned knowledge when solving a new task. An observation of these environments includes the position and velocity of the end-effector and object (if it exists) and the position of the goal. An action of these environments contains the position variation of the end-effector and the distance between the two fingers.

Setup. We compare the soft actor-critic (Haarnoja et al., 2018) algorithm with hindsight experience replay (Andrychowicz et al., 2017) under RL (SAC) and KGRL (SAC-KGRL) formulations. In SAC-KGRL, we add the learnable inner key, knowledge keys, and query on top of the actor model of SAC. We define the following two external knowledge policies for SAC-KGRL, where $\mathbf{p}_{ee} \in \mathbb{R}^3$ is the position of the end-effector and $\mathbf{p}_{obj} \in \mathbb{R}^3$ is the position of the object. (**KG1**) move straightly to the goal: if $\|\mathbf{p}_{ee} - \mathbf{p}_{obj}\|_2 < \varepsilon$, move straightly to the goal with the gripper closed; otherwise, stay unmoved. Note that there is no object in Reach, so ε is set to infinity in this environment, and **KG1** always guides the agent towards the goal. (**KG2, for Pick-And-Place only**) move straightly to the object: if $\|\mathbf{p}_{ee} - \mathbf{p}_{obj}\|_2 \geq \varepsilon$, move straightly to the object with the gripper opened; otherwise, stay unmoved. ε is set to 0.03 in Pick-And-Place.

Results. Figure 5 shows the training results of Reach and Pick-And-Place. In both environments, SAC-KGRL requires less samples to succeed and shows more stability. We also study the effects of different knowledge combinations in Pick-And-Place. The learning curves indicate that using both KG1 and KG2 leads to slightly better sample efficiency than using only one of them, but there is no significant difference among the three combinations.

Analysis of Generalizability. We analyze if our KGRL policies are more generalizable by training in both environments with a smaller goal range and testing with larger goal ranges. The training goal range is 0.1 times smaller than the original one, and the testing goal ranges are the original one multiplied by 0.1 to 1. Figure 6 shows the generalizability results. The success rate of SAC and

SAC-KGRL agents both decreases as the testing goal range increases, but the SAC-KGRL agent in Reach and the SAC-KGRL(all) agent in Pick-And-Place are less affected by the changes, especially when the goal ranges are larger. These results suggest that an KGRL agent has more potential to generalize to unseen environments.

Environment	Reach			Pick-And-Place				
Knowledge	All	IA	KG1	All	IA + KG1	IA + KG2	IA	KG1 + KG2
Success Rate	1	1	1	0.99	0.99	0.99	0.99	0.05

Table 2: Evaluation results of removing part of the knowledge set of our KGRL policies (IA: inner actor, All: IA + all KGs). Our proposed method enables the inner actor to *mimic* the strategies of external knowledge, and it may become a more powerful piece of new knowledge.

Analysis of Knowledge Control Flexibility. We demonstrate the flexibility of our method recomposing and rearranging knowledge by removing part of the knowledge set during evaluation. We evaluate Reach with the policies learned by SAC-KGRL and Pick-And-Place with the policies learned by SAC-KGRL(all). Table 2 shows the testing results of both tasks after removing part of the knowledge set. The learned inner actors in both environments achieve high success rate without the external guidance, indicating that a KGRL agent learns to not only *follow* the guidance but also *mimic* its strategy. This imitation process allows an agent to develop an inner policy that outperforms external strategies, and itself can then become new knowledge useful for solving other tasks.

Model	SAC	SAC-KGRL-all	SAC-KGRL-KG1	SAC-KGRL-KG2
100k Training Steps	0.24	0.56	0.25	0.48
200k Training Steps	0.61	0.99	0.99	0.98
Best	0.8	1	1	1

Table 3: Zero-shot transfer results of Pick-And-Place policies to Reach. Our proposed method is more effective and efficient in reusing learned knowledge.

Analysis of Knowledge Reusability. To demonstrate that our KGRL method enables better knowledge reusability, we apply zero-shot transfer from the learned Pick-And-Place policies to Reach. Note that the observation spaces of Reach and Pick-And-Place are different since there is no object in Reach. Hence, we apply zeros as the object's relative position and velocity in Reach to unify their observation spaces. Table 3 shows the success rate of different Pick-And-Place policies applied to Reach. The results indicate that SAC-KGRL is more efficient in reusing knowledge since the transfers are mostly successful after only 200k training steps.

6 Conclusion and Discussion

In this work, we introduce KGRL, a novel RL problem discussing how an agent can follow an arbitrary set of external guidance and learn its own policy. Also, the agent should be able to repeatedly reuse previously learned knowledge and dynamically recompose and rearrange knowledge to adapt to environmental changes. To address these challenges, we propose a simple yet effective actor model that performs an embedding-based attention mechanism. The experimental results show that the proposed model achieves the goal of KGRL. Our KGRL algorithm also satisfies the four properties of efficient learning (Kaelbling, 2020) and learns interpretable agent behaviors.

This work serves as an initial effort for the ultimate goal of KGRL, which is to automatically collect a large group of knowledge from different sources, such as books, the Internet, or previous training processes. This group of knowledge can be shared among multiple environments/tasks and continuously expanded, and a wide variety of artificial agents can use it as their guidance. Moving forward, the following research directions in KGRL are worth exploring: (1) incorporating restrictions that should be strictly followed during the learning and inference stages, (2) composing knowledge with different state and action spaces, and (3) dealing with more complex relationships among different knowledge, such as conditional dependence and conflicts. We hope this work can motivate future research in KGRL.

REFERENCES

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. PMLR, 2017.

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Haitham Bou Ammar, Rasul Tutunov, and Eric Eaton. Safe policy search for lifelong reinforcement learning with sublinear regret. In *International Conference on Machine Learning*, pages 2361–2369. PMLR, 2015.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. *Advances in neural information processing systems*, 30, 2017.

Aude G Billard, Sylvain Calinon, and Rüdiger Dillmann. Learning from humans. *Springer handbook of robotics*, pages 1995–2014, 2016.

Ethan Brooks, Janarthanan Rajendran, Richard L Lewis, and Satinder Singh. Reinforcement learning of implicit and explicit control flow instructions. In *International Conference on Machine Learning*, pages 1082–1091. PMLR, 2021.

Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3387–3395, 2019.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid`, 2018.

Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. *Advances in neural information processing systems*, 5, 1992.

Dongsheng Ding, Xiaohan Wei, Zhuoran Yang, Zhaoran Wang, and Mihailo Jovanovic. Provably efficient safe exploration via primal-dual policy optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 3304–3312. PMLR, 2021.

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

Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. *Advances in neural information processing systems*, 30, 2017.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. Meta learning shared hierarchies. *arXiv preprint arXiv:1710.09767*, 2017.

Javier Garcia and Fernando Fernández. Safe exploration of state and action spaces in reinforcement learning. *Journal of Artificial Intelligence Research*, 45:515–564, 2012.

Ruben Glatt, Felipe Leno Da Silva, and Anna Helena Reali Costa. Towards knowledge transfer in deep reinforcement learning. In *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 91–96. IEEE, 2016.

Vinicius G Goecks, Gregory M Gremillion, Vernon J Lawhern, John Valasek, and Nicholas R Waytowich. Integrating behavior cloning and reinforcement learning for improved performance in dense and sparse reward environments. *arXiv preprint arXiv:1910.04281*, 2019.

Abhishek Gupta, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. Unsupervised meta-learning for reinforcement learning. *arXiv preprint arXiv:1806.04640*, 2018a.

Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. *Advances in neural information processing systems*, 31, 2018b.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John Agapiou, et al. Learning from demonstrations for real world reinforcement learning. 2017.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Yiding Jiang, Shixiang Shane Gu, Kevin P Murphy, and Chelsea Finn. Language as an abstraction for hierarchical deep reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Leslie Pack Kaelbling. The foundation of efficient robot learning. *Science*, 369(6506):915–916, 2020.

Khimya Khetarpal and Doina Precup. Learning options with interest functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9955–9956, 2019.

Jaekyeom Kim, Seohong Park, and Gunhee Kim. Unsupervised skill discovery with bottleneck option learning. *arXiv preprint arXiv:2106.14305*, 2021.

George Konidaris and Andrew Barto. Autonomous shaping: Knowledge transfer in reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 489–496, 2006.

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. *arXiv preprint arXiv:1712.00948*, 2017.

Hao Liu, Richard Socher, and Caiming Xiong. Taming maml: Efficient unbiased meta-reinforcement learning. In *International conference on machine learning*, pages 4061–4071. PMLR, 2019.

Puze Liu, Davide Tateo, Haitham Bou Ammar, and Jan Peters. Robot reinforcement learning on the constraint manifold. In *Conference on Robot Learning*, pages 1357–1366. PMLR, 2022.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning. *arXiv preprint arXiv:1810.01257*, 2018a.

Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018b.

Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018.

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6292–6299. IEEE, 2018.

Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel Van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)*, 37(4):1–14, 2018a.

Xue Bin Peng, Angjoo Kanazawa, Sam Toyer, Pieter Abbeel, and Sergey Levine. Variational discriminator bottleneck: Improving imitation learning, inverse rl, and gans by constraining information flow. *arXiv preprint arXiv:1810.00821*, 2018b.

Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. Mcp: Learning composable hierarchical control with multiplicative compositional policies. *Advances in Neural Information Processing Systems*, 32, 2019.

Mark Pfeiffer, Samarth Shukla, Matteo Turchetta, Cesar Cadena, Andreas Krause, Roland Siegwart, and Juan Nieto. Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations. *IEEE Robotics and Automation Letters*, 3(4):4423–4430, 2018.

Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

Ahmed H Qureshi, Jacob J Johnson, Yuzhe Qin, Taylor Henderson, Byron Boots, and Michael C Yip. Composing task-agnostic policies with deep reinforcement learning. *arXiv preprint arXiv:1905.10681*, 2019.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22 (268):1–8, 2021. URL `http://jmlr.org/papers/v22/20-1364.html`.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

Matthew Riemer, Miao Liu, and Gerald Tesauro. Learning abstract options. *Advances in neural information processing systems*, 31, 2018.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Martin Stolle and Doina Precup. Learning options in reinforcement learning. In *International Symposium on abstraction, reformulation, and approximation*, pages 212–223. Springer, 2002.

Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning*, pages 9133–9143. PMLR, 2020.

Shao-Hua Sun, Te-Lin Wu, and Joseph J Lim. Program guided agent. In *International Conference on Learning Representations*, 2019.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Yunzhe Tao, Sahika Genc, Jonathan Chung, Tao Sun, and Sunil Mallya. Repaint: Knowledge transfer in deep reinforcement learning. In *International Conference on Machine Learning*, pages 10141–10152. PMLR, 2021.

Chen Tessler, Shahar Givony, Tom Zahavy, Daniel Mankowitz, and Shie Mannor. A deep hierarchical approach to lifelong learning in minecraft. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Brijen Thananjeyan, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minho Hwang, Joseph E Gonzalez, Julian Ibarz, Chelsea Finn, and Ken Goldberg. Recovery rl: Safe reinforcement learning with learned recovery zones. *IEEE Robotics and Automation Letters*, 6(3):4915–4922, 2021.

Dweep Trivedi, Jesse Zhang, Shao-Hua Sun, and Joseph J Lim. Learning to synthesize programs as interpretable and generalizable policies. *Advances in Neural Information Processing Systems*, 34:25146–25163, 2021.

Wei-Cheng Tseng, Jin-Siang Lin, Yao-Min Feng, and Min Sun. Toward robust long range policy transfer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9958–9966, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 3540–3549. PMLR, 2017.

Zhiyuan Xu, Kun Wu, Zhengping Che, Jian Tang, and Jieping Ye. Knowledge transfer in multi-task deep reinforcement learning for continuous control. *Advances in Neural Information Processing Systems*, 33: 15146–15155, 2020.

Yichen Yang, Jeevana Priya Inala, Osbert Bastani, Yewen Pu, Armando Solar-Lezama, and Martin Rinard. Program synthesis guided reinforcement learning for partially observed environments. *Advances in Neural Information Processing Systems*, 34, 2021.

Haiyan Yin and Sinno Jialin Pan. Knowledge transfer for deep reinforcement learning with hierarchical experience replay. In *Thirty-First AAAI conference on artificial intelligence*, 2017.

Ruohan Zhang, Faraz Torabi, Lin Guan, Dana H Ballard, and Peter Stone. Leveraging human guidance for deep reinforcement learning tasks. *arXiv preprint arXiv:1909.09906*, 2019.

Shangtong Zhang and Shimon Whiteson. Dac: The double actor-critic architecture for learning options. *Advances in Neural Information Processing Systems*, 32, 2019.

Zelin Zhao, Karan Samel, Binghong Chen, et al. Proto: Program-guided transformer for program-guided tasks. *Advances in Neural Information Processing Systems*, 34, 2021.

A EXPERIMENTAL DETAILS

A.1 MINIGRID ENVIRONMENTS

A.1.1 ENVIRONMENTAL DETAILS

The environmental details can be found at `https://github.com/maximecb/gym-minigrid`.

A.1.2 MODEL ARCHITECTURE

We use the same model architecture for all the six environments. Each external knowledge embedding is generated by the PyTorch Paszke et al. (2019) module, `nn.Embedding`, with $d_k = 8$. The inner actor network, ϕ_{in}, is a 3-layer convolutional neural network followed by a 1-layer multi-layer perceptron (MLP). Its detailed architecture can be found in `https://github.com/lcswillems/rl-starter-files`. The inner key and query networks, $\phi_{\mathbf{k}}$ and $\phi_{\mathbf{u}}$, share the same MLP layer with ϕ_{in}, and each of them is followed by Tanh activation and an output layer with the dimension being d_k.

A.1.3 HYPERPARAMETERS

Each training trial in Section 5.1 is run with 5 different random seeds, and we report the average performance of the 5 trained agents for analyses. For the underlying PPO algorithm, we follow the implementation of `https://github.com/lcswillems/rl-starter-files` with unchanged hyperparameters.

A.2 OPENAI ROBOTIC ENVIRONMENTS

A.2.1 ENVIRONMENTAL DETAILS

A state $\mathbf{s}_t$ in unmodified Reach, where $\mathbf{s}_t \in \mathbb{R}^{10}$, contains the position and velocity of the end-effector, the distance between the two grippers, and their velocity. A state $\mathbf{s}_t$ in Pick-And-Place, where $\mathbf{s}_t \in \mathbb{R}^{25}$, contains the same information as that in Reach, the position, rotation, and velocity of the object, and the relative position between the object and the end-effector. In modified Reach, which is used in the analysis of knowledge reusability, a state $\mathbf{s}_t$, where $\mathbf{s}_t \in \mathbb{R}^{25}$, contains the same information as that in Pick-And-Place, but the object's information, except its relative position, is set to 0.

An action $\mathbf{a}_t$ in all environments, where $\mathbf{a}_t \in \mathbb{R}^4$, contains the position variation of the end-effector and the distance between the two grippers.

A.2.2 MODEL ARCHITECTURE

Each external knowledge embedding is generated by the PyTorch module, `nn.Embedding`, with $d_k = 4$ for both Reach and Pick-And-Place. The inner actor network, ϕ_{in}, is an MLP for both tasks. ϕ_{in} in Reach is with two hidden layers and a hidden size of 64 units, and that in Pick-And-Place is with three hidden layers and a hidden size of 512 units. The critic network, ϕ_c, is with the same architecture as that of ϕ_{in}. The inner key network, $\phi_{\mathbf{k}}$, is an MLP for both tasks. Different from the inner key network in MiniGrid environments, $\phi_{\mathbf{k}}$ in OpenAI robotic environments does not share layers with ϕ_{in}. $\phi_{\mathbf{k}}$ in Reach is with one hidden layer and a hidden size of 32 units, and that in Pick-And-Place is with two hidden layers and a hidden size of 64 units. The query network, $\phi_{\mathbf{u}}$, is with the same architecture as that of $\phi_{\mathbf{k}}$. All hidden layers in $\phi_{in}, \phi_c, \phi_{\mathbf{k}}$, and $\phi_{\mathbf{u}}$ are followed by ReLU activation.

A.2.3 HYPERPARAMETERS

Each training trial in Section 5.2 is run with 5 different random seeds, and each testing trial is run with 100 different random initializations. For the underlying SAC algorithm in Section 5.2, we follow the implementation of Stable-Baselines3 (SB3) Raffin et al. (2021). The training timesteps for Reach and Pick-And-Place are 20K and 1M respectively. The learning rates are 10^{-3} and 5×10^{-4} respectively. The batch sizes are 256 and 2048 respectively. The sizes of the replay buffer are both 1M. The discount factors γ are both 0.95. The temperature of entropy is adjusted automatically in both environments as described in Haarnoja et al. (2018).