Autonomous Agents and Multi-Agent Systems (2020) 34:28
https://doi.org/10.1007/s10458-020-09451-0

Model primitives for hierarchical lifelong reinforcement learning

Bohan Wu[1] · **Jayesh K. Gupta**[2] · **Mykel Kochenderfer**[2]

Abstract
Learning interpretable and transferable subpolicies and performing task decomposition from a single, complex task is difficult. Such decomposition can lead to immense sample efficiency gains in lifelong learning. Some traditional hierarchical reinforcement learning techniques enforce this decomposition in a top-down manner, while meta-learning techniques require a task distribution at hand to learn such decompositions. This article presents a framework for using diverse suboptimal world models to decompose complex task solutions into simpler modular subpolicies. Given these world models, this framework performs decomposition of a single source task in a bottom up manner, concurrently learning the required modular subpolicies as well as a controller to coordinate them. We perform a series of experiments on high dimensional continuous action control tasks to demonstrate the effectiveness of this approach at both complex single-task learning and lifelong learning. Finally, we perform ablation studies to understand the importance and robustness of different elements in the framework and limitations to this approach.

Keywords Reinforcement learning · Task decomposition · Transfer · Lifelong learning · Hierarchical learning

1 Introduction

Lifelong learning [67] is the ability of a system to continuously learn from data, building on what has been previously learned. In the context of reinforcement learning, we want a lifelong learning agent to continue solving a *series* of related tasks drawn from a task distribution rather than a single, isolated task. Agents should be able to *transfer* knowledge

✉ Jayesh K. Gupta
jkg@cs.stanford.edu

Bohan Wu
bohan.wu@columbia.edu

Mykel Kochenderfer
mykel@stanford.edu

[1] Columbia University, New York, USA

[2] Stanford University, Stanford, USA

Published online: 25 February 2020

gained in previous tasks to improve performance on future tasks. This setting is different from multi-task reinforcement learning [63, 65, 72] and various meta-reinforcement learning settings [3, 18, 19, 21, 27, 73], where the agent is jointly trained on multiple task environments. Not only do such non-incremental settings make the problem of discovering common structures between tasks easier, they also allow the methods to ignore the problem of catastrophic forgetting [46], which is the inability to solve previous tasks after learning to solve new tasks in a sequential learning setting. We refer the reader to Parisi et al. [50] for an extensive review.

Our work takes a step towards solutions for such incremental, sequential settings [66]. We draw on the idea of modularity [48], i.e. complex behavior is often built from the composition of simpler building blocks. While learning to perform a complex task, we force the agent to perform task decomposition: breaking its solution down into simpler subpolicies instead of learning a single monolithic policy. This task decomposition allows our agent to rapidly learn another related task by transferring subpolicies learned from previous tasks. We hypothesize that many complex tasks are heavily structured and hierarchical in nature. The likelihood and quality of transfer of an agent's solution increases if it can capture such shared structure.

Recent works have visited the problem of task decomposition from a variety of viewpoints. For example, to reduce the sample complexity of task decomposition, the ideas of state abstraction [1, 2, 4, 5, 12, 28, 36, 41] and state aggregation [9, 47, 51] are introduced to compress the original state representation into one that eases learning of decomposition. Moreover, Isele and Cosgun [33] introduced selective memory storage and distribution matching to prevent catastrophic forgetting. Finally, Isele et al. [34] leverages the availability of high-level task descriptors to achieve zero-shot lifelong learning transfer.

A key ingredient of our proposal is the idea of world models [29, 37, 40]—transition models that can predict future sensory data given the agent's current actions and observations. Often these can be learned in a semi-supervised [29] or supervised [49] manner. The world however is complex, and learning models that are consistent enough to plan with is not only hard [62], but planning with such one-step models is also suboptimal [32]. We posit that the requirement that these world models be good predictors of the world state is unnecessary, provided we have a multiplicity of such models. We use the term *model primitives* to refer to these suboptimal world models. Since each model primitive is only relatively better at predicting the next states within a certain region of the environment space, we call this area the model primitive's *region of specialization*.

Model primitives allow the agent to decompose the task being performed into subtasks according to their regions of specialization and learn a specialized subpolicy for each of these regions or subtasks. The same model primitives are used to learn a gating controller to improve, adapt and compose the various subpolicies to solve a given task in a manner similar to a mixture of experts framework [45].

Our framework assumes that at least a subset of model primitives are useful across a range of tasks and environments. This assumption is less restrictive than that of successor representations [8, 14, 24, 39, 43, 44, 74, 76]. Even though successor representations decouple the state transitions from the rewards (representing the tasks or goals), the transitions learned are policy dependent and can only transfer across tasks that share the same environment dynamics.

There are alternative approaches to learning hierarchical spatio-temporal decompositions from the rewards seen while interacting with the environment, with Options [61] being the most widely applied framework to formalize the notion of a subpolicy in a sequential decision making process. Such approaches include the option-critic architecture [6] and

follow-on works [16, 31, 35, 42, 69, 75] that allow learning such decompositions in a single task environment. However, this method requires regularization hyperparameters that are tricky to set. As observed by Vezhnevets et al. [71], its learning often collapses to a single subpolicy. Other strategies require hand-designed features [20] or priors that promote diversity [13, 17] for hierarchical learning.

Moreover, we posit that capturing the shared structure across task-environments can be more useful in the context of transfer for lifelong learning than reward-based task specific structures. First, using sub-options for task decomposition requires gradient signals from rewards, which could be less stable due to sparsity. If sub-options are frozen during task decomposition, then they must be near-optimal, which model primitives need not be. If sub-options are not frozen and open for learning, then at the beginning of training, the higher-level controller is not accurately performing task decomposition, resulting in the wrong gradients being back-propagated into the sub-options, causing the sub-options themselves to change and no longer being near-optimal in their own specific tasks. In short, sub-option hierarchical learning is subject to the nonstationary bidirectional interplay between sub-options and task decomposition. During learning, either one can affect the other. In comparison, the task composition based on model primitives is achieved through supervised learning and independent of subpolicy performance. The interplay is unidirectional—the gating controller is affected by model primitives, but the model primitives remain unchanged throughout lifelong learning. Second, model primitives are sub-optimal and need not be accurate as we will demonstrate later in this work. In comparison, the provided sub-options need to be near-optimal for effective task decomposition. Third, model primitives are not specific to behaviors, but specific to environments, while sub-options are behavior-specific. Future work can build on our work to explore how model primitives can be utilized to decompose different behaviors and subtasks within the same local environments.

Other approaches include meta-learning algorithms such as Meta-Adaptation [3], Meta-Critic [59] and Meta-Learning Shared Hierarchies (MLSH) [21], which require a multiplicity of pretrained subpolicies or joint training on related tasks, which limits their ability to perform well in the sequential lifelong learning setting. While not in the field of reinforcement learning, Shamwell et al. [57] leveraged deep neural networks to learn a multiplicity of world models simultaneously.

To summarize our contributions:

- Given diverse suboptimal world models that can be practically learned, we propose an effective method to uncover a task decomposition encoded in these models.
- We propose an architecture to jointly train all decomposed subpolicies and a gating controller to solve a given source task and achieve significant sample efficiency improvements in solving target tasks in lifelong learning.
- We demonstrate the effectiveness of this approach at both single-task and lifelong learning in complex domains with high-dimensional observations and continuous actions.

2 Preliminaries

We assume the standard reinforcement learning (RL) formulation: an *agent* interacts with an *environment* to maximize the expected reward [60]. The environment is modeled as a Markov decision process (MDP), which is defined by $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$ with a

state space $\mathcal{S}$, an action space $\mathcal{A}$, a reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, a dynamics model $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Pi(\mathcal{S})$, and a discount factor $\gamma \in [0, 1]$. Here, $\Pi(\cdot)$ defines a probability distribution over a set. The agent acts according to stationary stochastic policies $\pi : \mathcal{S} \to \Pi(\mathcal{A})$, which specify action choice probabilities for each state. Each policy π has a corresponding $Q_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ function that defines the expected discounted cumulative reward for taking an action a from state s and following the policy π from that point onward.

2.1 Lifelong reinforcement learning

In a lifelong learning setting, the agent must sequentially interact with multiple tasks and successfully solve each of them. Let N denote the total number of tasks to learn. Adopting the framework from Brunskill and Li [11], in lifelong RL, the agent receives $\mathcal{S}, \mathcal{A}$, initial state distribution $\rho_0 \in \Pi(\mathcal{S})$, horizon H, discount factor γ, and an unknown distribution over reward-transition function pairs, D. The agent samples $(\mathcal{R}_i, \mathcal{T}_i) \sim D$ and interacts with the MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}_i, \mathcal{T}_i, \gamma \rangle$ for a maximum of H timesteps, starting according to the initial state distribution ρ_0, where $i \in \{1, \ldots, N\}$. After solving the given MDP or after H timesteps, whichever occurs first, the agent resamples from D and repeats.

2.2 Decomposed representation

The fundamental question in lifelong learning is to determine what knowledge should be captured by the agent from the tasks it has already solved so that it can improve its performance on future tasks. When learning with functional approximation, this translates to learning the right representation—the one with the right inductive bias for the tasks in the distribution. Given the assumption that the set of related tasks for lifelong learning share a lot of structure, the ideal representation should be able to capture this shared structure.

Thrun and Pratt [68] summarized various representation decomposition methods into two major categories. Modern approaches to avoiding catastrophic forgetting during transfer tend to fall into either category. The first category partitions the parameter space into task-specific parameters and general parameters [54]. The second category learns constraints that can be superimposed when learning a new function [38].

A popular approach within the first category is to use what Thrun and Pratt [68] term as *recursive* functional decomposition. This approach assumes that the task solution can be decomposed into a function of the form $f_i = h_i \circ g$, where h_i is task-specific whereas g is the same for all f_i and $i \in \{1, \ldots, N\}$. This scheme has been particularly effective in computer vision where early convolutional layers in deep convolutional networks trained on ImageNet [15, 58] become a very effective g for a variety of tasks. However, this approach to decomposition often fails in DeepRL because of two main reasons. First, the gradients used to train such networks are noisier as a result of Monte Carlo sampling. Second, the i.i.d. assumption for training data often fails.

We instead focus on devising an effective *piecewise* functional decomposition of the parameter space, as defined by Thrun and Pratt [68]. The assumption behind this decomposition is that each function f_i can be represented by a collection of functions $h_1, \ldots, h_K$, where $K \ll N$, and N is the number of tasks to learn. Our hypothesis is that this type of decomposition is much more effective and easier to learn in RL.

3 Model primitive hierarchical reinforcement learning

This section outlines the Model Primitive Hierarchical Reinforcement Learning (MPHRL) framework (Fig. 1) to address the problem of effective piecewise functional decomposition for transfer across a distribution of tasks.

3.1 Model primitives and gating

The key assumption in MPHRL is access to several diverse world models of the environment dynamics. These models can be seen as instances of learned approximations to the true environment dynamics $\mathcal{T}$. In reality, these dynamics can even be non-stationary. Therefore, the task of learning a complete model of the environment dynamics might be too difficult. Instead, it can be much easier to train multiple approximate models that specialize in different parts of the environment. We use the term *model primitives* to refer to these approximate world models.

The number of model primitives K depends mainly on the specific environment and domain. As demonstrated in Sect. 4.3.3, this quantity K need not be exact for MPHRL to achieve significant sample efficiency improvement in lifelong learning, but a larger number of diverse model primitives (over-specification of regions of specialization) is more favorable than a smaller number of diverse model primitives (under-specification of regions of specialization).

Suppose we have access to K model primitives: $\hat{\mathcal{T}}_k : \mathcal{S} \times \mathcal{A} \to \Pi(\mathcal{S})$, where $k \in \{1, \ldots, K\}$. For simplicity, we can assign a label M_k to each $\hat{\mathcal{T}}_k$, such that their predictions of the environment's transition probabilities can be denoted by $\hat{\mathcal{T}}(s_{t+1} \mid s_t, a_t, M_k)$.

3.1.1 Subpolicies

The goal of the MPHRL framework is to use these suboptimal predictions from different model primitives to decompose the task space into their individual regions of specialization, and learn different subpolicies $\pi_k : \mathcal{S} \to \Pi(\mathcal{A})$ that can focus on these regions. In the function approximation regime, each subpolicy π_k belongs to a fixed class of smoothly parameterized stochastic policies $\{\pi_{\theta_k} \mid \theta_k \in \Theta\}$, where Θ is a set of valid parameter vectors.

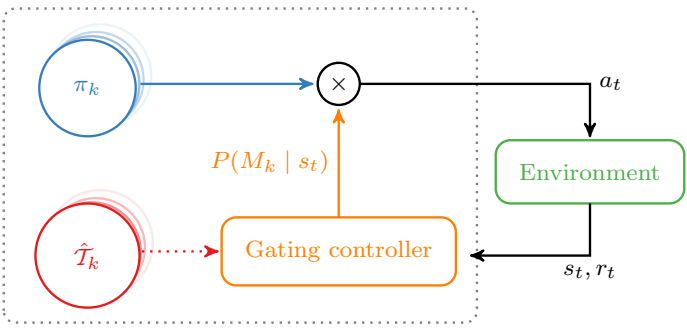


Fig. 1 Diagram of MPHRL Architecture. Solid arrows are active during both learning and execution. Dotted arrows are active only during learning. Here, $k \in \{1, \ldots, K\}$, where K is the number of model primitives the lifelong learning agent has access to

Model primitives are suboptimal and make incorrect predictions about the next state. Therefore we do not use them for planning or model-based learning of subpolicies directly. In other words, the subpolicies do not receive any information about the dynamics from the model primitives. The model primitives only help task decomposition, but do not provide any information to each subpolicy as to how to solve the subtask. Instead, model primitives give rise to useful functional decompositions and allow subpolicies to be learned in a model-free way. This also means that the sample efficiency improvement achieved by MPHRL does actually stem from the use of model primitives, and not from the fact that the model primitives encode domain knowledge about the optimal sub-behaviors. Model primitives are sub-optimal, which makes the domain knowledge themselves inadequate for robust subpolicy learning.

3.1.2 Gating controller

Taking inspiration from the mixture-of-experts literature [45], where the output from multiple experts can be combined using probabilistic gating functions, MPHRL decomposes the solution for a given task into multiple "expert" subpolicies and a gating controller that can compose them to solve the task. We want this switching behavior to be probabilistic and continuous, as opposed to one-hot hard selections, to avoid abrupt transitions. During learning, we want this controller to help assign the reward signal to the correct blend of subpolicies to ensure effective learning as well as decomposition.

Model primitives can give the gating controller access to the optimal mixture of subpolicies to activate and use, but do not tell the subpolicies how to solve each subtask, since the subpolicies are learned via model-free RL.

To begin, we observe that the diverse model primitives divide the world states into multiple regions of specialization that have different transition dynamics. Local environments with different transition dynamics can require different sets of desirable behaviors to navigate effectively. It is natural to reason that a separate subpolicy should be learned for each region of specialization, so that these subpolicies will be less identical, therefore making hierarchical lifelong RL more sample-efficient. This means that an effective task decomposition should divide the world states precisely into these regions of specialization and learn a subpolicy for each region. Therefore, our goal is to accurately categorize each world state s_t into the correct region of the specialization.

We achieve this goal by computing $P(M_k \mid s_t, a_t, s_{t+1})$. This quantity denotes the ground truth probability of the kth model primitive being the most accurate predictor of the current transition (s_t, a_t, s_{t+1}). This quantity also denotes the probability of s_t belonging to the kth region of specialization. Here, to say that a model primitive specializes in the current state is to say that out of all model primitives, this model primitive makes the most accurate prediction of the current transition (s_t, a_t, s_{t+1}). This is also analogous to stating that this model primitive has the lowest inverse expected divergence between the ground truth next state distribution $\mathcal{T}(\cdot \mid s_t, a_t)$ and its own predicted next state distribution $\hat{\mathcal{T}}(\cdot \mid s_t, a_t, M_k)$. By accurately computing $P(M_k \mid s_t, a_t, s_{t+1})$, the agent will be able to learn a dedicated subpolicy for each region of specialization, decompose the task effectively, and finally achieve more sample-efficient lifelong RL.

To compute $P(M_k \mid s_t, a_t, s_{t+1})$, we note that using Bayes' rule:

$$P(M_k \mid s_t, a_t, s_{t+1}) \propto P(M_k \mid s_t)\pi_k(a_t \mid s_t)\hat{\mathcal{T}}(s_{t+1} \mid s_t, a_t, M_k) \tag{1}$$

because $\pi_k(a_t \mid s_t) = \pi(a_t \mid s_t, M_k)$.

Intuitively, $P(M_k \mid s_t, a_t, s_{t+1})$ should be higher for the kth model primitive if it is more accurate in predicting the current transition, or equivalently, if $\hat{\mathcal{T}}(s_{t+1} \mid s_t, a_t, M_k)$ is higher. Hence the inclusion of $\hat{\mathcal{T}}(s_{t+1} \mid s_t, a_t, M_k)$ in calculating $P(M_k \mid s_t, a_t, s_{t+1})$, as given by the Bayes' rule.

However, the agent only has access to the current state s_t during execution. That is, the agent's gating controller needs to correctly identify which of the K regions of specialization the current state s_t belongs to, without being able to observe a_t and s_{t+1} in advance. Therefore, the agent needs to marginalize out s_{t+1} and a_t such that the model choice only depends on the current state s_t:

$$P(M_k \mid s_t) = \int_{s_{t+1} \in \mathcal{S}} \int_{a_t \in \mathcal{A}} P(M_k \mid s_t, a_t, s_{t+1}) \pi(a_t \mid s_t) \mathcal{T}(s_{t+1} \mid s_t, a_t) da_t ds_{t+1} \tag{2}$$

This is equivalent to:

$$P(M_k \mid s_t) = \mathbb{E}_{a_t \sim \pi(\cdot \mid s_t), s_{t+1} \sim \mathcal{T}(\cdot \mid s_t, a_t)} \left[P(M_k \mid s_t, a_t, s_{t+1}) \right] \tag{3}$$

Unfortunately, computing these integrals not only requires a_t and s_{t+1}, but also uses expensive Monte Carlo methods. However, we can approximate $P(M_k \mid s_t)$ using discriminative learning [52].

Concretely, we parameterize the gating controller (GC) as a categorical distribution $P_\phi(M_k \mid s_t) = P(M_k \mid s_t; \phi)$ and learn ϕ by minimizing the conditional cross entropy loss between $\mathbb{E}_{a_t \sim \pi(\cdot \mid s_t), s_{t+1} \sim \mathcal{T}(\cdot \mid s_t, a_t)} \left[P(M_k \mid s_t, a_t, s_{t+1}) \right]$ and $P_\phi(M_k \mid s_t)$ for all sampled transitions (s_t, a_t, s_{t+1}) in a rollout:

$$\underset{\phi}{\text{minimize}} \; \mathcal{L}^{GC} \tag{4}$$

where

$$\mathcal{L}^{GC} = \sum_{s_t} \sum_{k} - \left(\sum_{s_{t+1}} \sum_{a_t} P(M_k \mid s_t, a_t, s_{t+1}) \right) \times \log P(M_k \mid s_t; \phi) \tag{5}$$

This is equivalent to an implicit Monte Carlo integration to compute the marginal if $a_t \sim \pi(\cdot \mid s_t), s_{t+1} \sim \mathcal{T}(\cdot \mid s_t, a_t)$. Although we cannot query $\pi(a_t \mid s_t)$ or $\mathcal{T}(s_{t+1} \mid s_t, a_t)$ directly, s_t, a_t, and s_{t+1} can be sampled according to their respective distributions while we perform rollouts in the environment. Despite the introduced bias in our estimates, we find Eq. 4 with Eq. 5 sufficient for effectively uncovering a task decomposition encoded in the provided primitives that would later lead to sample efficiency improvement in lifelong learning.

3.1.3 Subpolicy composition

Taking inspiration from mixture-of-experts, the gating controller composes the subpolicies into a mixture policy:

$$\pi(a_t \mid s_t) = \sum_{k=1}^{K} P_\phi(M_k \mid s_t) \pi_k(a_t \mid s_t) \tag{6}$$

See Fig. 2 for a concrete example.

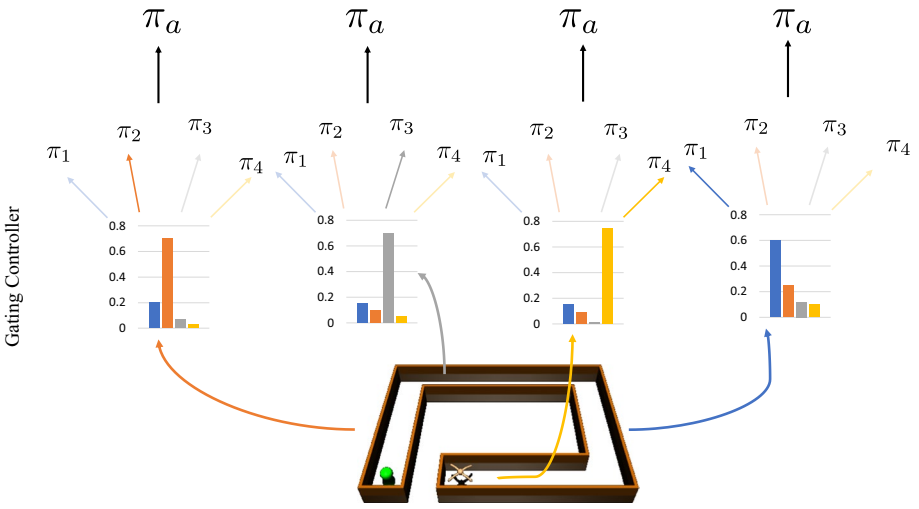


Fig. 2 Visualization of the policy composition using the Gating Controller's output in a Maze environment, where the amber four-legged robot is tasked to reach the green goal. The subpolicies $\pi_{1,2,3,4}$ specialize in N, S, W, E corridors respectively (Color figure online)

3.1.4 Decoupling cross entropy from action distribution

During a rollout, the agent samples a_t and s_{t+1} as follows:

$$a_t \sim \pi(\cdot \mid s_t) \tag{7}$$

$$s_{t+1} \sim \mathcal{T}(\cdot \mid s_t, a_t) \tag{8}$$

The π_k from Eq. 1 gets coupled with this sampling distribution, making the target distribution in Eq. 5 no longer stationary and the approximation process empirically difficult, as demonstrated in Sect. 4.3.10. We alleviate this issue by approximating $\pi_k(a_t \mid s_t)$ with $\pi(a_t \mid s_t)$, effectively treating the action distribution as a distribution independent of k and as a result, making the target distribution stationary. This transforms Eq. 1 into:

$$\begin{aligned} \hat{P}(M_k \mid s_t, a_t, s_{t+1}) &\propto P(M_k \mid s_t)\pi_k(a_t \mid s_t)\hat{\mathcal{T}}(s_{t+1} \mid s_t, a_t, M_k) \\ &\approx P(M_k \mid s_t)\pi(a_t \mid s_t)\hat{\mathcal{T}}(s_{t+1} \mid s_t, a_t, M_k) \\ &\propto P(M_k \mid s_t)\hat{\mathcal{T}}(s_{t+1} \mid s_t, a_t, M_k) \end{aligned} \tag{9}$$

An alternative view of this same mathematical relationship is given by:

$$\hat{P}(M_k \mid s_t, a_t, s_{t+1}) \propto P(M_k \mid s_t, a_t)\hat{\mathcal{T}}(s_{t+1} \mid s_t, a_t, M_k) \tag{10}$$

As the gating controller becomes increasingly discriminative and accurate, $P(M_k \mid s_t) \approx P(M_k \mid s_t, a_t)$ since the action a_t will not further inform the correct task decomposition. As a result:

$$\begin{aligned}\hat{P}(M_k \mid s_t, a_t, s_{t+1}) &\propto P(M_k \mid s_t, a_t)\hat{\mathcal{T}}(s_{t+1} \mid s_t, a_t, M_k) \\ &\approx P(M_k \mid s_t)\hat{\mathcal{T}}(s_{t+1} \mid s_t, a_t, M_k)\end{aligned} \tag{11}$$

3.2 Learning

Since the focus of this work is on difficult continuous state and action problems, we mostly concentrate on the issue of policy optimization and how it integrates with the gating controller. The standard policy (SP) optimization objective is:

$$\underset{\theta}{\text{maximize}}\ \mathcal{L}^{SP} = \mathbb{E}_{\rho_0,\pi_\theta}[\pi_\theta(a_t \mid s_t)Q_{\pi_\theta}(s_t, a_t)] \tag{12}$$

With baseline subtraction for variance reduction, this turns into [55]:

$$\underset{\theta}{\text{maximize}}\ \mathcal{L}^{PG} = \mathbb{E}_{\rho_0,\pi_\theta}[\pi_\theta(a_t \mid s_t)\hat{A}_t] \tag{13}$$

where $\hat{A}_t$ is an estimator of the advantage function [7].

In MPHRL, we directly use the mixture policy as defined by Eq. 6. The standard policy gradients (PG) get weighted by the probability outputs of the gating controller, enforcing the required specialization by factorizing into:

$$\hat{g}_k = \mathbb{E}_{\rho_0,\pi_{\theta_k}}\left[P_\phi(M_k \mid s_t)\nabla_{\theta_k} \log \pi_{\theta_k}(a_t \mid s_t)\hat{A}_t\right] \tag{14}$$

In practice, we use the Clipped PPO objective [56] instead to perform stable updates by limiting the step size. This includes adding a baseline estimator (BL) parameterized by ψ for value prediction and variance reduction. We optimize ψ according to the following loss:

$$\mathcal{L}^{BL} = \mathbb{E}\left[\left\| V_\psi - V_{\pi_\theta} \right\|^2\right] \tag{15}$$

We summarize this single-task learning algorithm in Algorithm 1, which results in a set of decomposed subpolicies, $\pi_{\theta_1}, \ldots, \pi_{\theta_K}$, and a gating controller P_ϕ that can modulate between them to solve the task under consideration.

Algorithm 1 MPHRL: single-task learning

1: Initialize $P_\phi, \pi_\theta = \{\pi_{\theta_1}, \ldots, \pi_{\theta_K}\}$, V_ψ
2: **while** not converged
3: Rollout trajectories $\tau \sim \pi_{\theta,\phi}$
4: Compute advantage estimates $\hat{A}_\tau$
5: Optimize $\mathcal{L}^{PG}$ wrt $\theta_1, \ldots, \theta_K$ with expectations taken over τ
6: Optimize $\mathcal{L}^{BL}$ wrt ψ with expectations taken over τ
7: Optimize $\mathcal{L}^{GC}$ wrt ϕ with expectations taken over τ

Lifelong learning: We have shown how MPHRL can decompose a single complex task solution into different functional components. Complex tasks often share structure and can be decomposed into similar sets of subtasks. Different tasks however require *different* recomposition of *similar* subtasks. Therefore, we transfer the subpolicies to learn target tasks, but not the gating controller or the baseline estimator. We summarize the lifelong learning algorithm in Algorithm 2, with the global variable `RESET` set to true.

Algorithm 2 MPHRL: Lifelong learning

Initialize P_ϕ, $\pi_\theta = \{\pi_{\theta_1}, \ldots, \pi_{\theta_K}\}, V_\psi$
for Tasks $(\mathcal{R}_i, \mathcal{T}_i) \sim D$
  if RESET
    Initialize P_ϕ, V_ψ
  while not converged
    Rollout trajectories $\tau \sim \pi_{\theta,\phi}$
    Compute advantage estimates $\hat{A}_\tau$
    Optimize $\mathcal{L}^{PG}$ wrt $\theta_1, \ldots, \theta_K$ with expectations taken over τ
    Optimize $\mathcal{L}^{BL}$ wrt ψ with expectations taken over τ
    Optimize $\mathcal{L}^{GC}$ wrt ϕ with expectations taken over τ

4 Experiments

Our experiments aim to answer two questions: (a) can model primitives ensure task decomposition? (b) does such decomposition improve transfer for lifelong learning?

We evaluate our approach in two challenging domains: a MuJoCo [70] ant navigating different mazes and a Stacker [64] arm picking up, placing and stacking different boxes. In our experiments, we use subpolicies that have Gaussian action distributions, with mean given by a multi-layer perceptron taking observations as input and standard deviations given by a different set of parameters. MPHRL's gating controller outputs a categorical distribution and is parameterized by another multi-layer perceptron. We also use a separate multi-layer perceptron for the baseline estimator. We use the standard clipped PPO [56] algorithm as a baseline to compare against MPHRL. Transferring network weights empirically led to worse performance for standard clipped PPO. Hence, we re-initialize the baseline PPO's weights for every task. For fair comparison, we also shrink the hidden layer size of MPHRL's subpolicy networks from 64 to 16 (as detailed in Table 1). We conduct each experiment across 5 different seeds and report performance means and standard deviations. Error bars in all performance figures below represent the standard deviation from the mean.

The focus of this work is on understanding the usefulness of model primitives for task decomposition and the resulting improvement in sample efficiency from transfer. To conduct perfectly controlled experiments with reliable and interpretable results, we first conduct experiments with model primitives created using the true next state provided by the environment simulator, while later in Sect. 4.3.7 we present lifelong learning experiments with non-hand-crafted, learned model primitives that exhibited similar sample efficiency improvement. Concretely, we apply distinct multivariate Gaussian noise models with covariance $\sigma\Sigma$ to the true next state. We then sample from this distribution to obtain the mean of the probability distribution of a model primitive's next state prediction, using Σ as its covariance. Here, σ is the noise scaling factor that distinguishes model primitives, while Σ refers to the empirical covariance of the sampled next states:

$$\mu \sim \mathcal{N}(s_{t+1}, \sigma_k \Sigma) \tag{16}$$

$$\hat{\mathcal{T}}(s_{t+1} \mid s_t, a_t, M_k) = \mathcal{N}(\mu, \Sigma) \tag{17}$$

Table 1 Hyperparameters: MPHRL and baseline PPO

Category	Hyper-parameter	Value
Number of model primitives: Maze	Single-task: L-Maze	2
	Single-task: D-Maze	4
	Lifelong learning: 10-Maze	4
	Ablation: H-V	2
	Ablation: velocity	2
	Ablation: extra	5
Number of model primitives: 8-Pickup&Place	lifelong learning: 8-Pickup&Place	12
	Ablation: Box	2
	Ablation: Action	6
Gating controller: network	Hidden layers	2
	Hidden dimension	64
Gating controller: base learning rate	Single/source Task[a] (10-Maze)	1×10^{-3}
	Single/source task (8-Pickup&Place)	3×10^{-2}
	Target tasks	3×10^{-3}
Gating controller: number of epoches/batch	Single/source task	1
	Target tasks	10
Baseline and model primitive networks[b]	Hidden layers	2
	Hidden dimension	64
	Base learning rate	3×10^{-4}
Subpolicy networks[c]	Hidden layers	2
	Hidden dimension (MPHRL)	16
	Hidden dimension (PPO)	64
	Base learning rate	3×10^{-4}
Optimization	Number of actors (10-Maze)	16
	Number of actors (8-Pickup&Place)	24
	Batch size/actor (10-Maze)	2048
	Batch size/actor (8-Pickup&Place)	1536
	Max. timesteps/task	3×10^{7}
	Minibatch size/actor	256
	Number of epoches/batch[d]	10
	Discount (γ)	0.99
	GAE parameter (λ)	0.95
	PPO clipping coeff. (ϵ)	0.2
	Gradient clipping	None
	VF coeff. (c_1)	1.0
	Entropy coeff. (c_2)	0
	Optimizer	Adam

[a]Single task refers to L-Maze and D-Maze; source and target tasks refer to the first task and all subsequent tasks in a lifelong learning taskset, respectively

[b]Baseline network hyperparameters apply to both MPHRL and baseline PPO; model primitive networks are for experiments with learned model primitives only

[c]The baseline PPO has no subpolicies, so the subpolicy network *is* the policy network

[d]Baseline and subpolicy networks only

Using Σ as opposed to a constant covariance is essential for controlled experiments because different elements of the observation space have different orders of magnitude. Sampling μ from a distribution effectively adds random bias to the model primitive's next state probability distribution.

Later on we relax these controlled experimental conditions and show how these model primitives can be trained in practice and how our framework can leverage such learned model primitives for task decomposition to achieve similar lifelong learning performance and sample efficiency improvement.

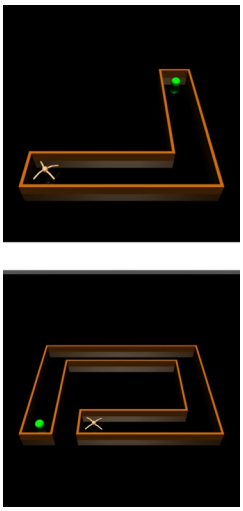
Our implementation is available at http://github.com/sisl/MPHRL, and the details for MPHRL and baseline PPO hyperparameters are presented in Table 1. Agent videos are available at 10-Maze Playlist URL and 8-Pickup&Place Playlist URL.

4.1 Single-task learning

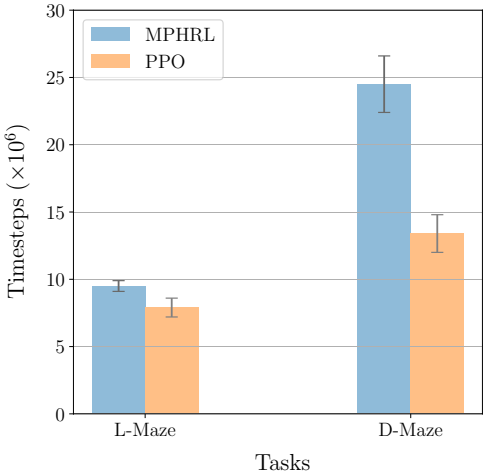
First, we focus on two single-task learning experiments where MPHRL learns a number of interpretable subpolicies to solve a single task. Both the L-Maze and D-Maze (Fig. 3a) tasks require the ant to learn to walk and reach the green goal within a finite horizon. For both tasks, both the goal and the initial ant locations are fixed.

4.1.1 Observation space

The observation space for both the L-Maze and D-Maze includes the standard joint angles and velocities, lidar information that tracks distances from walls on each side, and the Manhattan distance to the goal.



(a) L-Maze (top) and D-Maze (bottom)



(b) Performance

Fig. 3 Single-task learning

4.1.2 Full details of reward structure

The reward at any given timesteps is composed of the forward reward, the control cost and the contact cost:

$$r_t = r_t^{forward} - c_t^{control} - c_t^{contact} \tag{18}$$

The forward reward is proportional to the difference in the (x, y) Euclidean distance to the next subgoal between the current and the previous timestep:

$$r_t^{forward} = 50 \times (d_{t-1} - d_t) \tag{19}$$

where

$$d_t = \left\| [x_t^{ant} - x_t^{subgoal}, y_t^{ant} - y_t^{subgoal}] \right\|_2 \tag{20}$$

The subgoals are located at the end of each sub-corridor.

The contact and control costs remain the same as the original OpenAI Gym [10]:

$$c_t^{control} = 0.5 \times \|a_t\|^2 \tag{21}$$

$$c_t^{contact} = 0.005 \times \|external\_contact\_forces\|^2 \tag{22}$$

.

4.1.3 Model primitives

For the L-Maze, the agent has access to two model primitives, one specializing in the horizontal (E, W) corridor and the other specializing in the vertical (N, S) corridor of the maze. Similarly for the D-Maze, the agent has access to four model primitives, one specializing in each N, S, E, W corridor of the maze. In their specialized corridors, the noise scaling factor $\sigma = 0$. Outside of their regions of specialization, $\sigma = 0.5$. Note that we set $\sigma = 0$ purely for the purpose of analysis.

4.1.4 Results and analysis

Section 4.3.7 shows that MPHRL is robust to learned, non-hand-crafted model primitives. Figure 3b shows the experimental results on these environments. Notice that using model primitives can increase the sample complexity on a single task. This is expected, since we are forcing the agent to decompose the solution for potential lifelong learning, which could be unnecessary for a single task. However, we will observe in the following section that this decomposition can lead to remarkable performance improvements in sequential knowledge transfer during lifelong learning.

4.2 Lifelong learning

To evaluate our framework's performance at lifelong learning, we introduce two tasksets: 10-Maze and 8-Pickup&Place. In our experiments, we report number of timesteps to reach target average success rate of 80% for 10-Maze tasks and 75% for 8-Pickup&Place tasks.

This is computed as the moving average of the percentages of success across 10 trailing iterations. During each iteration, each actor generates a complete trajectory that either succeeds or fails, and the percentage of success for this iteration is then computed by averaging across all actors.

4.2.1 10-Maze

To evaluate MPHRL's performance in lifelong learning, we generate a family of 10 random mazes for the MuJoCo [70] Ant environment, referred to as the 10-Maze lifelong learning taskset (Fig. 4) hereafter. The goal, the observation space, the Gaussian noise models, and the model primitives remain the same as in the D-Maze environment in Sect. 4.1. The full reward structure is also the same as "D-Maze", as detailed in Sect. 4.1.2. Each task is considered successful when the ant is within a very small distance to the green goal. Otherwise, the task is considered a failure. The agent has a maximum of 3×10^7 timesteps to reach 80% success rate in each of the 10 tasks. As shown in Fig. 5 and Table 2, MPHRL requires nearly double the number of timesteps to learn the useful, decomposed subpolicies (Fig. 6) in the first task. However, this cost gets amortized over the entire taskset, with MPHRL taking half the total number of timesteps of the baseline PPO, exhibiting strong subpolicy transfer.

4.2.2 8-Pickup&Place

We modify the Stacker task [64] to create the 8-Pickup&Place lifelong learning taskset. As shown in Fig. 7, a robotic arm is tasked to bring 2 boxes to their respective goal locations in a certain order. Marked by colors red, green, and blue, the goal locations reside within two short walls forming a "stack".

Each of the 8 tasks has a maximum of 3 goal locations. The observation space of the agent includes joint angles and velocities, box and goal locations, the boxes' relative distances to each other, and the current stage of the task encoded as one-hot vectors. The goal locations, the starting position and orientation of the robot, and the locations of the boxes are all initialized randomly for every episode. Furthermore, in order to allow the

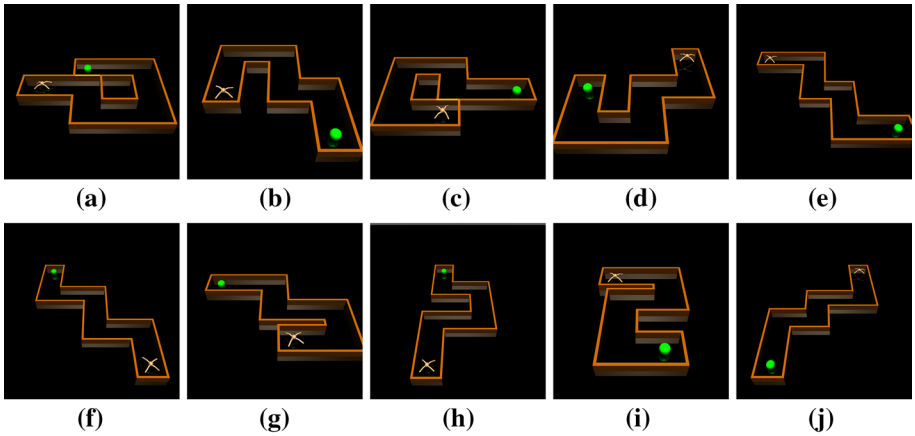


Fig. 4 The 10-Maze lifelong learning taskset

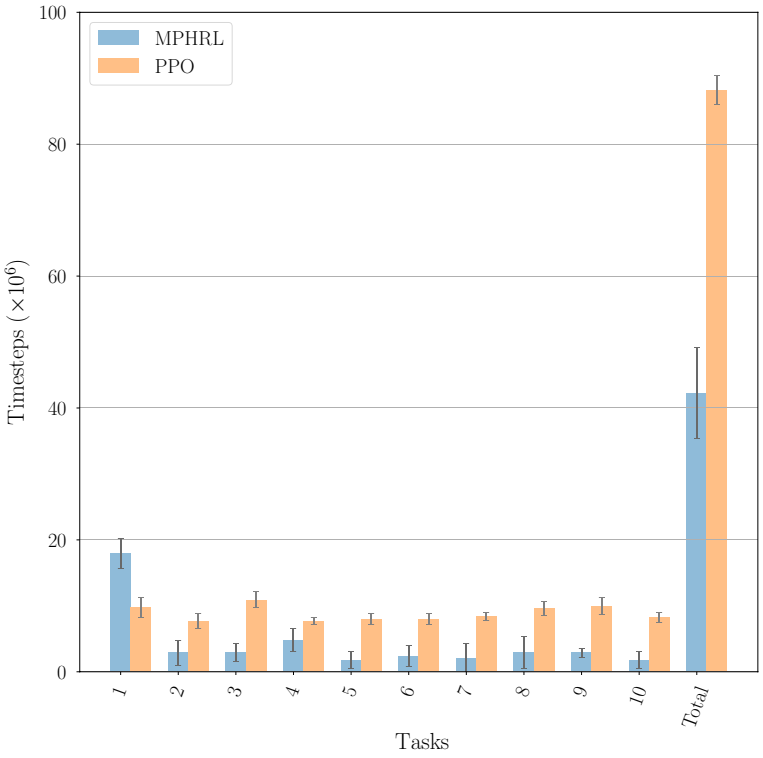


Fig. 5 10-Maze: MPHRL versus PPO for lifelong learning

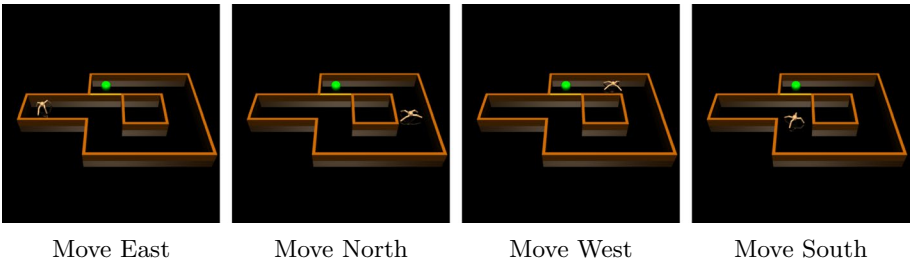


Fig. 6 4 decomposed subpolicies from the 10-Maze lifelong learning taskset

baseline PPO to solve the taskset more easily, episodes in 8-Pickup&Place tasks can sometimes be initialized to the state where some boxes are already in their goal locations. The probability of such initialization is constant with respect to the remaining goal locations to solve. For example, in Task (e) in Fig. 7, with $\frac{1}{3}$ probability an episode is initialized with no boxes already in their goal locations. With another $\frac{1}{3}$ probability, the episode is initialized with the black box already in the left stack. With the remaining $\frac{1}{3}$ probability, the episode is initialized with both the black and white boxes in the left stack (the white box will be on top of the black box), in which case all the robot has to

Table 2 10-Maze: MPHRL versus PPO for lifelong learning

Algorithm	Timesteps to reach 80% average success rate ($\times 10^6$)										
	1	2	3	4	5	6	7	8	9	10	Total
MPHRL	17.9 ± 2.3	2.9 ± 1.9	2.9 ± 1.4	4.8 ± 1.7	1.8 ± 1.3	2.4 ± 1.6	2.0 ± 2.3	2.9 ± 2.4	2.9 ± 0.7	1.8 ± 1.3	42.2 ± 6.9
PPO	9.8 ± 1.5	7.7 ± 1.2	10.9 ± 1.2	7.7 ± 0.5	8.0 ± 0.8	8.0 ± 0.8	8.4 ± 0.6	9.6 ± 1.0	10.0 ± 1.3	8.2 ± 0.8	88.2 ± 2.2

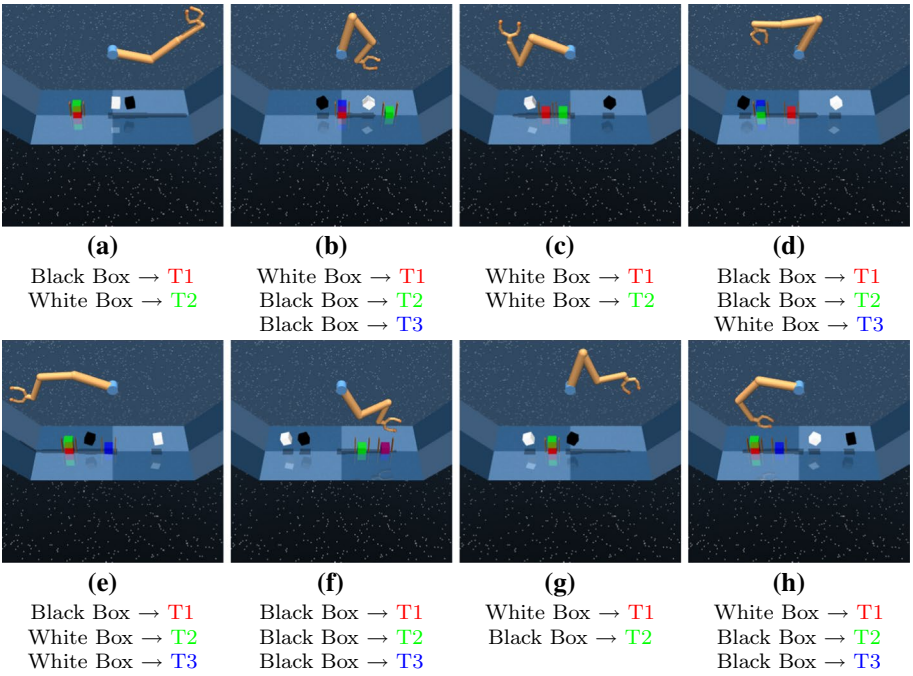


Fig. 7 The 8-Pickup&Place lifelong learning taskset. T1, T2, and T3 refer to Target Goal Location 1 (red), Target Goal Location 2 (green), and Target Goal Location 3 (blue) (Color figure online)

do to complete the task is to move the white box from the left stack to the right stack. When all boxes are in their correct goal locations within a very small distance, the episode is considered successful. Otherwise, the episode is considered a failure.

The reward at any timestep is calculated as the difference in distance to the next task subgoal:

$$r_t = d_{t-1} - d_t \tag{23}$$

where

$$d_t = \begin{cases} 10 \times \left\| \mathbf{p}_{grasp} - \mathbf{p}_{above\_box} \right\|_2 & \text{if subgoal is reach above} \\ 20 \times \left\| \mathbf{p}_{grasp} - \mathbf{p}_{box} \right\|_2 & \text{if subgoal is lower to} \\ 50 \times \left\| \mathbf{p}_{f1} - \mathbf{p}_{f2} \right\|_2 - 5 \times \left\| \mathbf{p}_{box} - \mathbf{p}_{target} \right\|_2 & \text{if subgoal is grasp} \\ 20 \times |y_{grasp} - y_{above\_target}| + \left\| \mathbf{p}_{f1} - \mathbf{p}_{f2} \right\|_2 & \text{if subgoal is pick up} \\ 10 \times \left\| \mathbf{p}_{grasp} - \mathbf{p}_{above\_target} \right\|_2 + \left\| \mathbf{p}_{f1} - \mathbf{p}_{f2} \right\|_2 & \text{if subgoal is carry} \\ 5 - 20 \times \left\| \mathbf{p}_{f1} - \mathbf{p}_{f2} \right\|_2 - 2 \times \left\| \mathbf{p}_{grasp} - \mathbf{p}_{box} \right\|_2 & \text{if subgoal is drop} \end{cases} \tag{24}$$

In Eq. 24, $f1$ refers to the left finger tip of the agent's gripper, $f2$ refers to the right finger tip of the agent's gripper, and $\mathbf{p}$ refers to the $[x, y]$ 2D position of the grasp, box, target, a point above the box, or a point above the target.

The agent has access to six model primitives for each box that specialize in approaching, lowering to, grasping, picking up, carrying, and dropping a certain box respectively. Similar to 10-Maze, model primitives have σ of 0 within their specialized stages and σ of 0.5 otherwise, purely for the purpose of analysis. Figure 8 and Table 3 show MPHRL's experimental performance by learning twelve useful subpolicies (Fig. 9) for this taskset. We notice again the strong transfer performance due to the decomposition forced by the model primitives. Note that this taskset is much more complex than 10-Maze such that MPHRL even accelerates the learning of the first task (4.0 ± 0.4 vs. 8.3 ± 4.2).

4.3 Ablation

We conduct ablation experiments to answer the following questions:

1. How much gain in sample efficiency is achieved by transferring subpolicies?
2. Can MPHRL learn good task decomposition even when the model primitives are quite noisy or when the source task does not cover all "cases" (partial decomposition)?
3. When does MPHRL fail to decompose the solution?
4. What kind of diversity in the model primitives is essential for performance?
5. When does MPHRL lead to negative transfer [23, 53]?

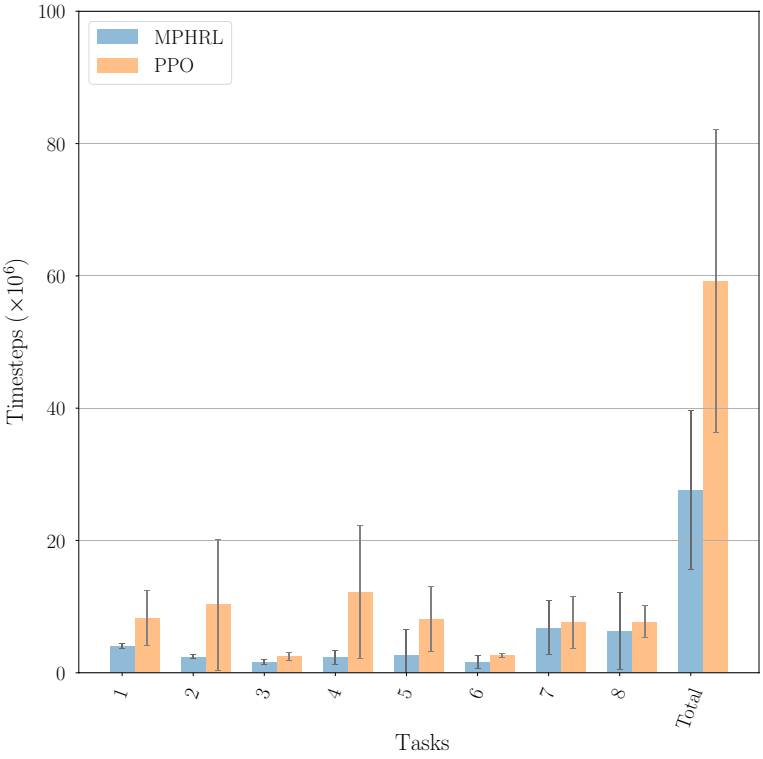


Fig. 8 8-Pickup&Place: MPHRL versus PPO for lifelong learning

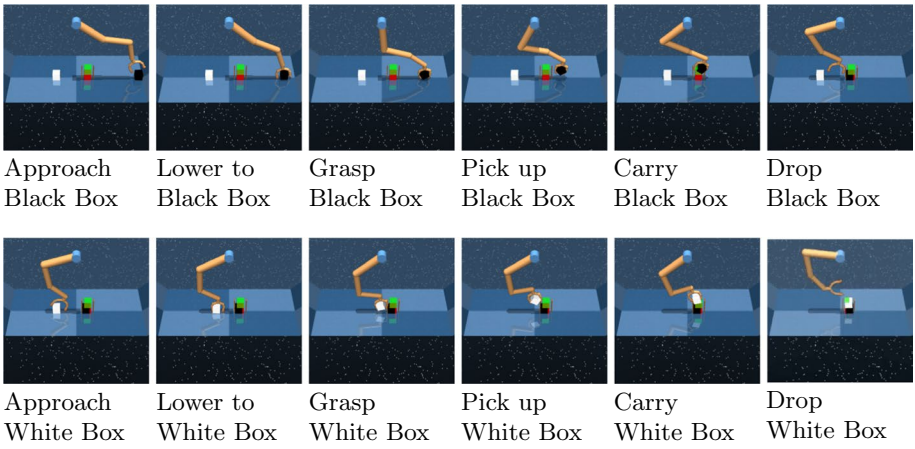


Fig. 9 12 decomposed subpolicies from the 8-Pickup&Place lifelong learning taskset

Table 3 8-Pickup&Place: MPHRL versus PPO for lifelong learning

Algorithm	Timesteps to reach 75% average success rate ($\times 10^6$)								
	1	2	3	4	5	6	7	8	Total
MPHRL	4.0 ± 0.4	2.4 ± 0.3	1.6 ± 0.4	2.3 ± 1.0	2.7 ± 3.8	1.6 ± 1.0	6.8 ± 4.1	6.3 ± 5.8	27.6 ± 12.0
PPO	8.3 ± 4.2	10.3 ± 9.9	2.5 ± 0.6	12.2 ± 10.0	8.1 ± 4.9	2.6 ± 0.3	7.6 ± 4.0	7.7 ± 2.4	59.2 ± 22.9

6. Is MPHRL's gain in sample efficiency a result of hand-crafted model primitives and how does it perform with non-hand-crafted, learned model primitives?

4.3.1 Model noise

MPHRL has the ability to decompose the solution even given bad model primitives. Since the learning is done model-free, these suboptimal model primitives should not strongly affect the learning performance so long as they remain sufficiently distinct. To investigate the limitations to this claim, we conduct five 10-Maze experiments using various sets of noisy model primitives. Below, the first value corresponds to the noise scaling factor σ *within* their individual regions of specialization, while the second value corresponds to σ *outside* of their regions of specialization.

(a) 0.4 and 0.5: good model primitives with limited distinction
(b) 0.5 and 1.0: good model primitives with reasonable distinction
(c) 5.0 and 10.0: bad model primitives with reasonable distinction
(d) 9.0 and 10.0: bad model primitives with limited distinction
(e) 0.5 and 0.5: good model primitives with no distinction

As shown in Fig. 10 and Table 4, (a), (b), (c), and (d) exhibit limited degradation in 10-Maze performance and still outperform baseline PPO (88.2 ± 2.2 as exhibited in Table 2). On the other hand, in (e) MPHRL took 22.0 ± 4.6 million timesteps to solve the

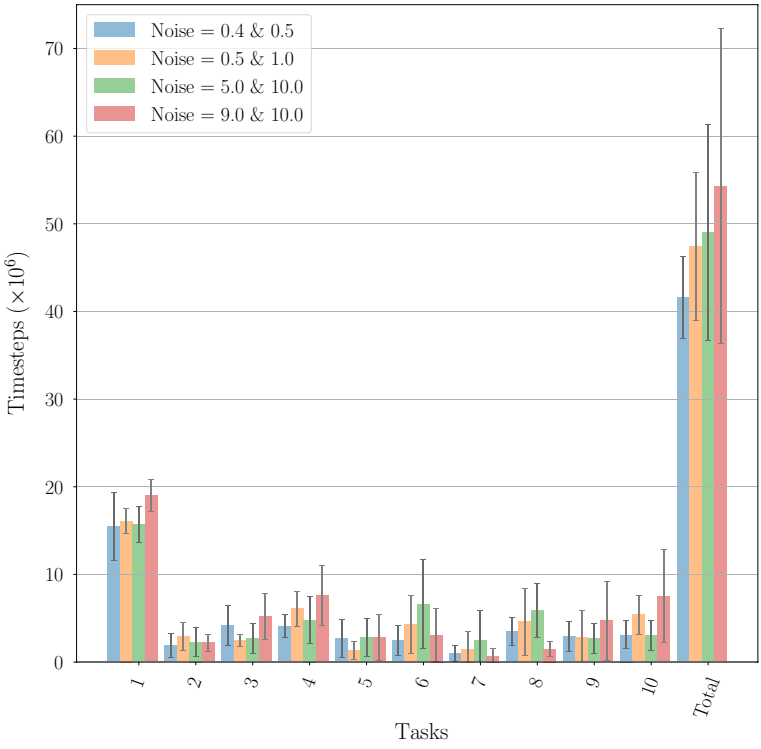


Fig. 10 10-Maze: effect of model noise on MPHRL sample complexity

first task and 2.8 ± 1.6 million timesteps to solve the second task, but failed to solve the third task within 30 million timesteps. This is because the model primitives are identical and provide no information about task decomposition. Figure 11 and Table 5 show similar results for 8-Pickup&Place, where bad, noisier model primitives (5 and 20) actually outperform good, less noisy (0 and 0.5) model primitives mainly due to a larger distinction among the model primitives. In summary, MPHRL is robust against bad model primitives so long as they maintain some relative distinction.

4.3.2 Overlapping model primitives

We next test the condition when there is substantial overlap in regions of specialization between different model primitives. For the 10-Maze taskset, the most plausible region for this confusion is at the corners. In this experiment, within each corner, the two model primitives whose specialized corridors share the corner have $\sigma = 0$ while the other two have $\sigma = 0.5$. Again, we set $\sigma = 0$ purely for analysis purpose, and Sect. 4.3.7 shows that MPHRL is robost to learned, non-hand-crafted model primitives. Figure 12 and Table 6 show the performance for model primitive confusion at the corners against the standard set of model primitives with no confusion. We observe that despite some performance degradation due to overlapping regions of specialization between the model primitives, MPHRL continues to outperform the PPO baseline (53.8 ± 10.6 vs. 88.2 ± 2.2).

Table 4 10-Maze: effect of model noise on MPHRL sample complexity

Noise		Timesteps to reach 80% average success rate ($\times 10^6$)										
T	F	1	2	3	4	5	6	7	8	9	10	Total
0.4	0.5	15.5 ± 3.9	1.9 ± 1.4	4.2 ± 2.3	4.1 ± 1.3	2.7 ± 2.2	2.5 ± 1.7	1.0 ± 0.9	3.5 ± 1.6	2.9 ± 1.7	3.1 ± 1.6	41.6 ± 4.7
0.5	1.0	16.1 ± 1.4	2.9 ± 1.6	2.5 ± 0.7	6.1 ± 2.0	1.3 ± 1.0	4.3 ± 3.3	1.5 ± 2.0	4.6 ± 3.8	2.8 ± 3.1	5.4 ± 2.2	47.4 ± 8.4
5	10	15.7 ± 2.1	2.3 ± 1.7	2.7 ± 1.7	4.8 ± 2.7	2.8 ± 2.2	6.6 ± 5.1	2.5 ± 3.4	5.9 ± 3.1	2.7 ± 1.7	3.0 ± 1.7	49.0 ± 12.3
9	10	19.0 ± 1.8	2.2 ± 1.0	5.2 ± 2.6	7.6 ± 3.4	2.8 ± 2.6	3.1 ± 3.0	0.7 ± 0.8	1.5 ± 0.9	4.7 ± 4.5	7.5 ± 5.3	54.3 ± 18.0
0.5	0.5	22.0 ± 4.6	2.8 ± 1.6	N/A								

Table 5 8-Pickup&Place: effect of model noise on MPHRL sample complexity

Noise		Timesteps to reach 75% average success rate ($\times 10^6$)								
T	F	1	2	3	4	5	6	7	8	Total
0	0.5	4.0 ± 0.4	2.4 ± 0.3	1.6 ± 0.4	2.3 ± 1.0	2.7 ± 3.8	1.6 ± 1.0	6.8 ± 4.1	6.3 ± 5.8	27.6 ± 12.0
5	20	4.5 ± 0.9	1.8 ± 0.2	1.7 ± 0.4	1.5 ± 0.6	1.0 ± 0.5	1.1 ± 0.5	2.2 ± 2.3	2.0 ± 2.3	15.8 ± 5.5

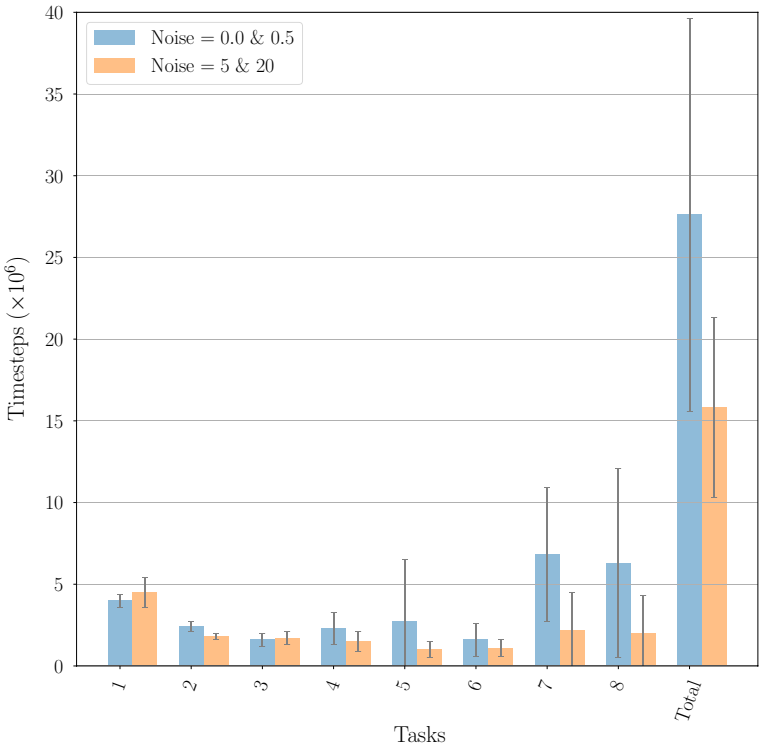


Fig. 11 8-Pickup&Place: effect of model noise on MPHRL sample complexity

4.3.3 Model diversity

Having tested MPHRL against model primitive noises and corner confusion, we experimented with sets of model primitives that are less desirable for 10-Maze:

(a) *Extra*: a fifth model primitive that specializes in states where the ant is moving horizontally; this is undesirable because this extra model primitive is redundant and can overlap with both the E and W model primitives.
(b) *H-V*: 2 model primitives specializing in horizontal (E, W) and vertical (N, S) corridors respectively; this is undesirable because such under-specification of the regions of spe-

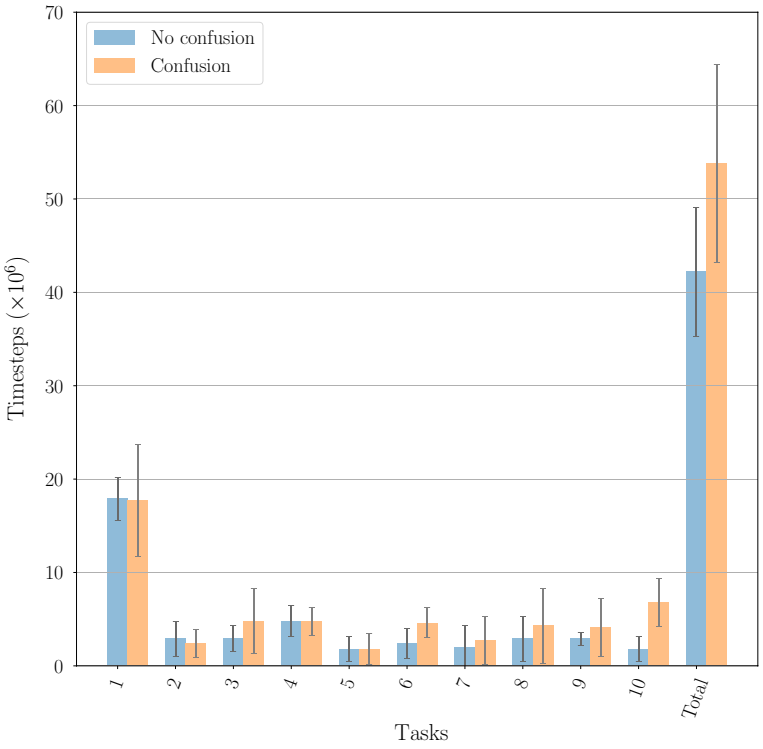


Fig. 12 10-Maze: Effect of model primitive confusion at the corners

cialization forces the ant agent to learn a single subpolicy for moving in both east and west corridors, and another subpolicy for moving in both north and south corridors.

(c) *Velocity*: 2 model primitives specializing in states where the ant agent is moving horizontally and vertically respectively; this is undesirable because such under-specification of the regions of specialization forces the ant agent to learn a single subpolicy for moving both east and west, and another subpolicy for moving both north and south.

and for 8-Pickup&Place:

(a) *Box*: 2 model primitives for all maneuvers on 2 boxes; this is undesirable because such under-specification of the regions of specialization forces the robot to learn one single subpolicy for all six different maneuvers on each box.

(b) *Action*: 6 model primitives for 6 maneuvers performed on boxes: approach, lower to, grasp, pick up, carry, and drop; this is undesirable because such under-specification of the regions of specialization forces the robot to learn one single subpolicy for each type of maneuver on all boxes.

Table 7 shows MPHRL is susceptible to some performance degradation given undesirable sets of model primitives. However, MPHRL still outperforms baseline PPO (50.7 ± 4.3

Table 6 10-Maze: effect of model primitive confusion at the corners

Noise			Timesteps to reach 80% average success rate ($\times 10^6$)										
T	F	Confusion	1	2	3	4	5	6	7	8	9	10	Total
0	0.5	Yes	17.7 ± 6.0	2.4 ± 1.5	4.8 ± 3.5	4.8 ± 1.5	1.8 ± 1.7	4.6 ± 1.6	2.7 ± 2.6	4.3 ± 4.0	4.1 ± 3.1	6.8 ± 2.6	53.8 ± 10.6
0	0.5	No	17.9 ± 2.3	2.9 ± 1.9	2.9 ± 1.4	4.8 ± 1.7	1.8 ± 1.3	2.4 ± 1.6	2.0 ± 2.3	2.9 ± 2.4	2.9 ± 0.7	1.8 ± 1.3	42.2 ± 6.9

Table 7 Effect of suboptimal model primitive types (N/A indicates failure to solve the task within 3×10^7 timesteps)

		Timesteps to reach target average success rate ($\times 10^6$)				
Taskset	MPs[a]	1	2	3	4	5
10-Maze	Extra	21.8 ± 1.5	4.3 ± 0.8	3.5 ± 0.9	5.2 ± 1.2	0.7 ± 0.5
10-Maze	H-V	28.1 ± 4.2	2.4 ± 0.6	18.9 ± 11.0	28.8 ± 2.8	N/A
10-Maze	Velocity	14.7 ± 2.4	1.4 ± 1.2	17.9 ± 11.2	20.5 ± 13.2	N/A
8-P&P[b]	Action	5.1 ± 0.6	4.8 ± 1.8	16.8 ± 12.6	26.9 ± 6.9	N/A
8-P&P	Box	18.6 ± 10.9	N/A			

		Timesteps to reach target average success rate ($\times 10^6$)					
Taskset	MPs	6	7	8	9	10	Total
10-Maze	Extra	3.6 ± 1.8	1.2 ± 1.1	1.5 ± 0.9	3.1 ± 0.7	5.8 ± 3.3	50.7 ± 4.3

[a]Model Primitives

[b]8-Pickup&Place

vs. 88.2 ± 2.2) when given an extra, undesirable model primitive. This indicates that for best transfer, the model primitives need to approximately capture the structure present in the taskset and avoid under-specification of regions of specialization.

4.3.4 Negative transfer and catastrophic forgetting

Lifelong learning agents with neural network function approximators face the problem of negative transfer and catastrophic forgetting. Negative transfer implies that the inductive bias learned by the agent on a subset of tasks during lifelong learning makes it worse at the full set of tasks. Ideally, the agent should find the solution quickly if the task has already been seen. More generally, given two sets of tasks T and T' such that $T \subset T'$, after being exposed to T' the agent should perform no worse (indicating no catastrophic forgetting), and preferably better (indicating positive transfer), than had it been exposed to T only.

In this 10-Maze experiment, we restore the subpolicy checkpoints after solving the 10 tasks and evaluate MPHRL's learning performance on the first 9 tasks. Similarly, we restore the subpolicy checkpoints after solving the first 6 tasks and evaluate MPHRL's performance on the first 5 tasks. The gating controller is reset for each task as in earlier experiments. We summarize the results in Table 8. MPHRL agents trained sequentially on 6 or 10 tasks quickly relearn the required behavior for all previously seen tasks, implying no catastrophic forgetting and that negative transfer is very limited with this approach. Moreover, if we compare the 10-task result to the 6-task result, we see remarkable improvements at transfer. This shows that MPHRL's robustness against catastrophic forgetting and negative transfer can increase with the number of seen tasks.

4.3.5 Oracle gating controller

One might suspect that all gains in sample efficiency come from hand-crafted model primitives because they allow the agent to learn a perfect gating controller. However, Fig. 13 shows that when the gating controller is already perfectly known, the reward curves for a 10-Maze experiment plateau at around 200, much lower than the required reward threshold of 800. As a result, this setup is unable to solve any 10-Maze tasks.

Since the 10-Maze taskset is composed of sequential subtasks, only one subpolicy will be learned in the first corridor when the gating controller is perfect. When transitioning to the second corridor, the second subpolicy needs to be learned from scratch, making the ant's subsequent rewards negative. This discourages the first subpolicy from entering the second corridor and activating the second subpolicy. Eventually, the ant stops moving forward close to the intersection between the first two corridors due to risk aversion.

More specifically, a random subpolicy could easily flip the ant agent upside down, after which the agent will no longer be able to move forward. As a result, the forward reward remains close to zero, while the control and contact costs continue to

Table 8 10-Maze: effect of experience

# Tasks	Timesteps to reach 80% average success rate ($\times 10^6$)								
	1	2	3	4	5	6	7	8	9
10	0.6 ± 0.1	0.4 ± 0.0	0.6 ± 0.0	0.5 ± 0.1	0.4 ± 0.0	2.3 ± 0.7	0.7 ± 0.2	2.6 ± 0.3	0.7 ± 0.1
6	3.1 ± 0.4	2.0 ± 0.2	3.5 ± 0.7	2.2 ± 0.7	1.8 ± 0.3				

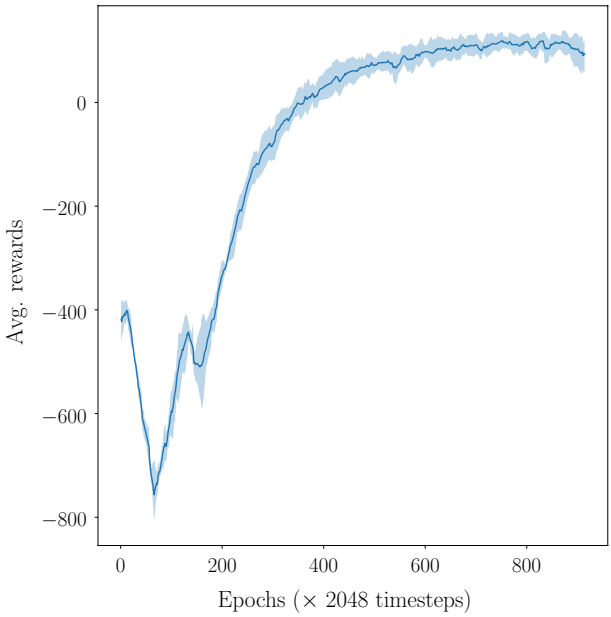


Fig. 13 Average rewards of MPHRL when using an oracle gating controller. The reward threshold for reaching 80% success rate for the first task is approximately 800

accumulate until episode termination. Indeed, if the two costs are eliminated, perhaps the first subpolicy will not prevent the agent from entering the second corridor and then the second subpolicy will train. Nevertheless, it will still take time for the each subsequent subpolicy to learn to prevent the ant from going upside down. In contrast, MPHRL's natural curriculum for gradual specialization allows multiple subpolicies to learn the basic skills for survival (not flipping the ant upside down) initially and simultaneously.

Here, we are providing a common issue found when using an oracle gating controller, where MPHRL's natural curriculum can help resolve. This issue is common in multi-stage tasks, particularly when the reward structure results in a random subpolicy in a subtask generating a negative cumulative reward. In this scenario, the subpolicy in the previous subtask actually learns to prevent the agent from entering the next subtask, causing globally suboptimal behavior.

4.3.6 Partial decomposition

To confirm that the ordering of tasks does not significantly affect MPHRL's performance, we modified the 10-Maze taskset to create the 10-Maze-v2 taskset (Fig. 14), in which the source task does not allow for complete decomposition into all useful subpolicies for the subsequent target tasks (there are no west corridors in the source task). The success rate threshold is at 70%. Again, we observe large improvement in sample efficiency over standard PPO (Fig. 15 and Table 9), demonstrating that MPHRL is robust against partial decomposition during lifelong learning.

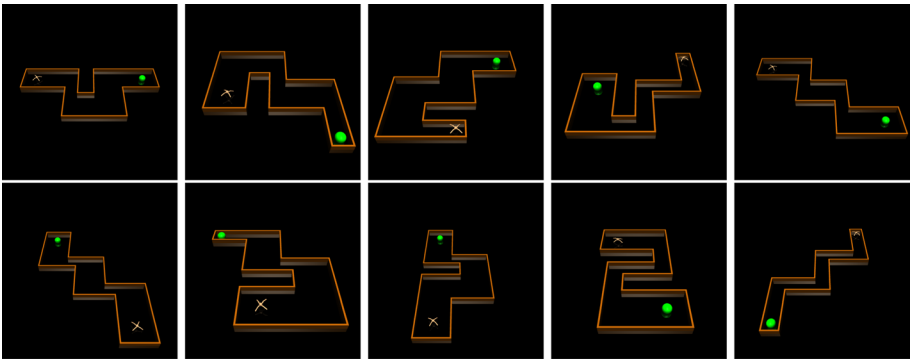


Fig. 14 The 10-Maze-v2 lifelong learning taskset

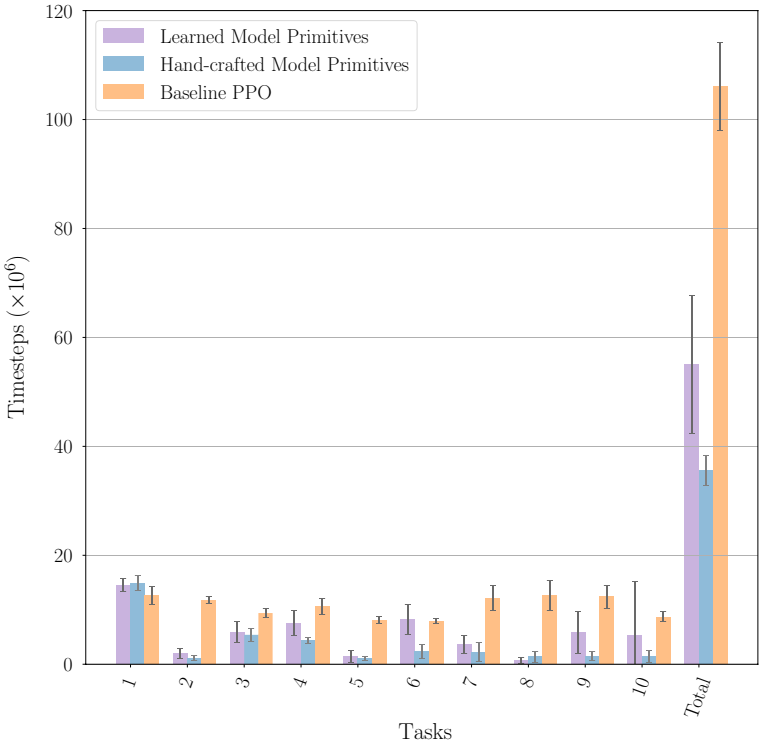


Fig. 15 10-Maze-v2: performance under partial decomposition and using learned model primitives

4.3.7 Learned model primitives

So far, this article has primarily focused on evaluating suboptimal world models for task decomposition in perfectly controlled experiments using hand-crafted model primitives. Now, we demonstrate that MPHRL is robust to learned, non-hand-crafted model

Table 9 10-Maze-v2: performance under partial decomposition and using learned model primitives

Algorithm	Timesteps to reach 70% average success rate ($\times 10^6$)										
	1	2	3	4	5	6	7	8	9	10	Total
MPHRL-C[a]	14.9 ± 1.4	1.1 ± 0.5	5.4 ± 1.2	4.4 ± 0.6	1.1 ± 0.4	2.3 ± 1.3	2.2 ± 1.7	1.4 ± 1.0	1.5 ± 0.9	1.4 ± 1.1	35.6 ± 2.8
MPHRL-L[b]	14.0 ± 1.5	2.5 ± 1.5	6.3 ± 1.9	7.3 ± 2.1	1.9 ± 1.4	9.3 ± 3.4	3.3 ± 1.6	0.7 ± 0.5	5.1 ± 3.7	4.7 ± 8.6	55.2 ± 11.0
PPO	12.6 ± 1.7	11.8 ± 0.6	9.4 ± 0.9	10.6 ± 1.4	8.1 ± 0.7	7.9 ± 0.5	12.2 ± 2.3	12.6 ± 2.8	12.4 ± 2.1	8.7 ± 0.9	106.1 ± 8.1

[a]MPHRL with hand-crafted model primitives

[b]MPHRL with learned model primitives

primitives. Here, we show one way to learn each model primitive for 10-Maze-v2 using three simple corridor environments demonstrated in Fig. 16. Concretely, we parameterize each model primitive using a multivariate Gaussian distribution. We learn the mean of this distribution via a multi-layer perceptron using a weighted mean square error in dynamics prediction as the training loss. The standard deviation is still derived from the empirical covariance Σ as described earlier. In this manner, model primitives can be learned straightforwardly via supervised learning. Even though the diversity in these learned model primitives is much more difficult to quantify and control than hand-crafted model primitives, their sample efficiency still substantially outperforms standard PPO and slightly underperforms hand-crafted model primitives with 0 and 0.5 model noises (Fig. 15 and Table 9).

Each model primitive took 3.3×10^6 timesteps to train, totaling 13.2×10^6 timesteps. Adding this cost to the 55.2×10^6 mean sample complexity of MPHRL using learned model primitives gives a total of $68.4 \pm 11.0 \times 10^6$ timesteps, which still significantly outperforms 106.1 ± 8.1 timesteps in the case of the baseline. In addition, the cost of learning model primitives is only a one-time cost (as opposed to a per-task cost) that gets amortized more significantly as the number of tasks increases, and here we have only used 10 tasks per lifelong learning experiment.

4.3.8 Gating controller transfer

Since the task decomposition varies across tasks, MPHRL transfers the subpolicies but not the gating controller during target task learning. To explore factors that could lead to negative transfer, we tested a version of MPHRL that does not re-initialize but rather transfers the gating controller in target tasks, as shown in Table 10 and Fig. 17 (in Gating and Subpolicies Transfer). Although the mean sample efficiency remains stable, its standard deviation increases, indicating volatility due to both positive and negative transfer. To avoid such volatility in transfer performance, MPHRL re-initializes the gating controller during target task learning.

4.3.9 Subpolicy transfer

To measure how much gain in sample efficiency MPHRL has achieved by transferring subpolicies alone, we conducted a 10-Maze experiment by re-initializing all network weights for every new task, essentially disabling any transfer. As shown in Table 10 and Fig. 17 (in No Gating or Subpolicies Transfer), sample complexity more than quintuples when

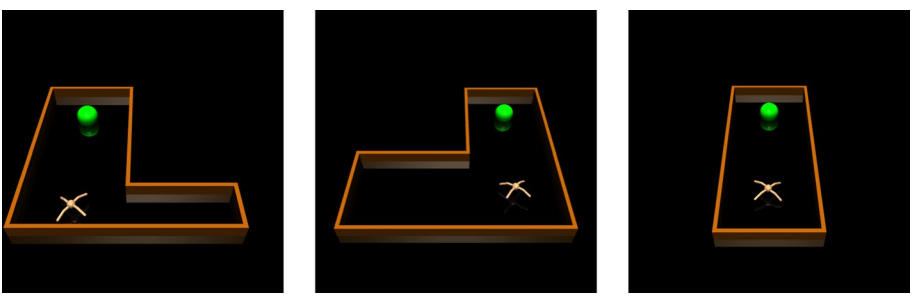


Fig. 16 Three simple corridor environments for learning the "N" model primitive

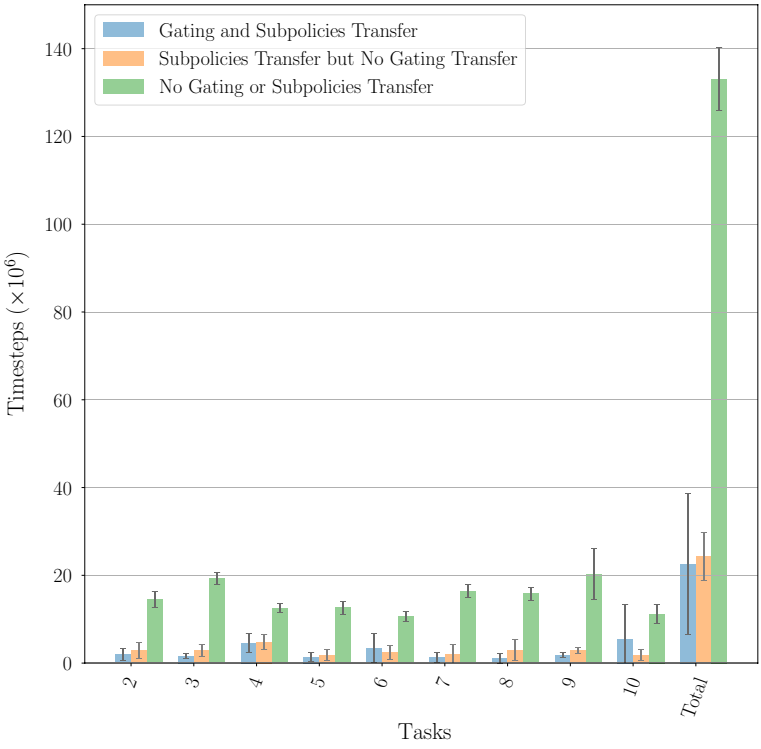


Fig. 17 10-Maze: gain in sample efficiency achieved by MPHRL's gating controller and subpolicies (target tasks performance only)

subpolicies are re-initialized. This demonstrates the significant sample efficiency improvement MPHRL achieves by transferring subpolicies.

4.3.10 Coupling between cross entropy and action distribution

To validate using $\hat{P}(M_k \mid s_t, a_t, s_{t+1})$ in Eq. 9 as opposed to $P(M_k \mid s_t, a_t, s_{t+1})$ from Eq. 1, we tested MPHRL with Eq. 1 on 10-Maze. All runs with different seeds failed to solve the first 5 tasks (Table 11). As the gating controller is re-initialized during transfer, most actions were chosen incorrectly. The gating controller is thus presented with the incorrect cross entropy target, which worsens the action distribution. The resulting vicious cycle forces the gating controller to converge to a suboptimal equilibrium against the incorrect target. Therefore, MPHRL adopts Eq. 9 rather than Eq. 1 to learn task decomposition.

5 Conclusions

In this paper, we showed how imperfect or even bad world models, i.e. model primitives, can be used to decompose a complex task into simpler subtasks. We introduced a framework that uses these model primitives to learn piecewise functional decompositions of solutions to complex tasks. The learned decomposed subpolicies can then be used to

Table 10 10-Maze: gain in sample efficiency achieved by MPHRL's gating controller and subpolicies (target tasks performance only)

Transfer		Timesteps to reach 80% average success rate ($\times 10^6$)									
GC[a]	Subpolicies	2	3	4	5	6	7	8	9	10	Total
Yes	Yes	1.9 ± 1.4	1.6 ± 0.5	4.6 ± 2.1	1.4 ± 1.1	3.4 ± 3.3	1.3 ± 1.2	1.0 ± 1.1	1.8 ± 0.6	5.5 ± 7.9	22.6 ± 16.1
No	Yes	2.9 ± 1.9	2.9 ± 1.4	4.8 ± 1.7	1.8 ± 1.3	2.4 ± 1.6	2.0 ± 2.3	2.9 ± 2.4	2.9 ± 0.7	1.8 ± 1.3	24.3 ± 5.4
No	No	14.5 ± 1.9	19.2 ± 1.4	12.5 ± 1.0	12.6 ± 1.5	10.6 ± 1.1	16.4 ± 1.5	15.8 ± 1.5	20.2 ± 5.8	11.2 ± 2.1	133.1 ± 7.1

[a]Gating controller

Table 11 10-Maze: effect of coupling between cross entropy and action distribution

Task	Timesteps to reach 80% average success rate ($\times 10^6$)				
	1	2	3	4	5
Timesteps ($\times 10^6$)	15.5 ± 1.2	3.4 ± 1.1	19.3 ± 8.0	28.9 ± 2.5	N/A

transfer to a variety of related tasks, reducing the overall lifelong learning sample complexity required to learn complex behaviors. Our experiments showed that such structured decomposition avoids negative transfer and catastrophic interference, two major concerns for lifelong learning systems. The experiments also demonstrated that MPHRL substantially improves lifelong learning sample efficiency with both learned and hand-crafted model primitives.

Our approach does not require access to accurate or hand-crafted model primitives. Neither does it need a well-designed task distribution (as in the case of meta-learning) or the incremental introduction of individual tasks. So long as the set of model primitives are sufficiently distinctive across the task distribution, MPHRL is robust to other imperfections.

6 Future work

6.1 Automatic task decomposition via online model primitive discovery

Learning useful and diverse model primitives, subpolicies and gating controller (i.e. task decomposition) *all simultaneously* is an important area for future work. Below we outline one idea that could make model-primitive-based automatic task decomposition possible.

Automatic discovery of model primitives, that is, without hand-crafted environment setup, means that no additional environments should be built solely for the purpose of learning model primitives. In other words, the only environment that model primitive learning can take place will be the source task environment itself. Under this scenario, model primitives, the gating controller and the subpolicies must be learned together simultaneously. From this perspective, the problem of automatic discovery of model primitives is the same as the problem of automatic task decomposition, or the challenge of simultaneously learning all three components of MPHRL.

Since the main requirement for a good set of model primitives is diversity, one can imagine the possibility of learning all three components of MPHRL at once online. At the beginning of training, it is unclear how many model primitives are needed, so the maximum number of model primitives allowed, $K_{\max}$, will be defined, e.g. $K_{\max} = 10$. On the other hand, it is initially unclear what the optimal number of model primitives (or subpolicies), K, will be. Therefore at the beginning of training, we assume that $K = 1$, and we increment K later during exploration when the agent realizes that more model primitives are needed. The key insight here is that in order to learn diverse model primitives, the data used to train these model primitives have to be diverse. Therefore, there needs to be an uncertainty threshold, β, below which a sample (s_t, a_t, r_t, s_{t+1}) will be allowed to be used to train a particular model primitive. In other words, each of the model primitives will only be trained on samples that the primitive itself finds not too surprising. A concrete instance of β for Gaussian model primitives can be the maximum mean square error between the ground truth next state and the predicted next state, below which the sample will be allowed to train the kth model primitive $\hat{\mathcal{T}}_k$:

$$\mathbb{1}\{(s_t, a_t, r_t, s_{t+1}) \text{ is allowed to train } \hat{\mathcal{T}}_k\} = \mathbb{1}\left\{ \left\| s_{t+1} - \hat{\mathcal{T}}_{\chi_k}(s_t, a_t) \right\|^2 \leq \beta \right\} \tag{25}$$

In the case where there are samples that no existing model primitive is familiar with (under the uncertainty threshold), a new model primitive will need to be trained to cover this new portion of the world states. In this case, we increment K and a new model primitive is added to the existing set. This process can repeat until MPHRL solves the entire source task, concurrently learning a set of model primitives, the same number of subpolicies and a gating controller. This idea can be summarized in Algorithm 3.

Using this algorithm to learn model primitives simultaneously does not necessarily result in learning suboptimal model primitives. In the 10-Maze environments, for example, this algorithm will not necessarily learn only 2 model primitives specializing in horizontal and vertical corridors, as opposed to learning 4 model primitives specializing in each of the N, E, S, W corridors. This is because given a good β hyperparameter, the existing model primitives will remain stable and untrained (e.g. E and N directions in the horizontal and vertical corridors respectively), while new model primitives are being added and trained (e.g. W and S directions in the horizontal and vertical corridors respectively). When entering the first W or S corridor, the E and N model primitives will find the W and S corridors unfamiliar in terms of next state transitions. When this unfamiliarity exceeds β, these unfamiliar transitions will not be allowed to train the E and N model primitives. In this case, Algorithm 3 will create new model primitives (i.e. the W and S model primitives) to handle the new W and S corridors. On the other hand, the sequential nature of Algorithm 3 ensures that E and N model primitives will be specialized enough such that transitions in the W and S corridors will appear unfamiliar to them.

Algorithm 3 Online MPHRL with Automatic Model Primitive Discovery

Initialize the gating controller P_ϕ, a set of subpolicies $\pi_\theta = \{\pi_{\theta_1}, \ldots, \pi_{\theta_{K_{max}}}\}$, a baseline network V_ψ, a set of model primitives $\hat{\mathcal{T}}_\chi = \{\hat{\mathcal{T}}_{\chi_1}, \ldots, \hat{\mathcal{T}}_{\chi_{K_{max}}}\}$, a model uncertainty threshold hyperparameter $\beta \in \mathbb{R}$, and $K = 1$
while not converged
  Rollout trajectories $\tau \sim \pi_{\theta,\phi}$
  Compute advantage estimates $\hat{A}_\tau$
  Divide τ into a number of sample subsets $\{\tau_1, \ldots, \tau_K\}$ such that all samples in τ_k has a model $\hat{\mathcal{T}}_{\chi_k}$ uncertainty lower than β, where $k = 1, \ldots, K$
  if some samples remain uncategorized
    Increment K
    Group all remaining samples into τ_K
  for $k = 1, \ldots, K$
    Optimize $\mathcal{L}^{MP} = \mathbb{E}\left[\left\| \mathcal{T} - \hat{\mathcal{T}}_{\chi_k} \right\|^2 \right]$ wrt χ_k with expectations taken over τ_k
  Optimize $\mathcal{L}^{PG}$ wrt $\theta_1, \ldots, \theta_{K_{max}}$ with expectations taken over τ
  Optimize $\mathcal{L}^{BL}$ wrt ψ with expectations taken over τ
  Optimize $\mathcal{L}^{GC}$ wrt ϕ with expectations taken over τ

6.2 Automatic task decomposition via neural processes and information bottlenecks

Given its ability to represent a distribution over stochastic processes, the recently introduced Neural Processes [22] can potentially be an efficient approach to build upon for model primitive learning. This approach can allow moving beyond simple discrete sets of model primitives to instead capturing the different uncertainties in a single model. It will also allow more complex gating controller than allowed by a simple categorical distribution. Moreover, change in uncertainties from low to high can lead to a better signal for learning a new model primitive.

Another interesting direction to take would be to achieve hierarchy through information-bottleneck [25, 30]. Modules specializing for different domains via different information bottlenecks could help fully automate the learning of task decomposition. An early attempt has been made in Goyal et al. [26] to set up an intrinsic competition mechanism to select the active subpolicy for a given input. Each subpolicy primitive has an information bottleneck on how much information it can use for a given input state. However, even for relatively simpler continuous control tasks, the policies required pre-training. Moreover, their objective is to achieve implicitly decentralized control. Still, similar ideas can also be applied in the context of model primitive learning. The K model primitives can be composed of an encoder $p_{\text{enc}}(Z_k \mid s_t, a_t)$ and a decoder $p_{\text{dec}}(s_{t+1} \mid Z_k)$. Here the output of the encoder, Z, captures the useful information in the current state and action and the decoder takes this encoded information to produce a distribution over next states. Together they can function as model primitives:

$$\hat{\mathcal{T}}(s_{t+1} \mid s_t, a_t, M_k) = \int_z p_{\text{enc}}(z_k \mid s_t, a_t) p_{\text{dec}}(s_{t+1} \mid z_k) dz_k \tag{26}$$

with information bottleneck enforced by penalizing KL divergence of Z and a prior. This however does not necessarily lead to specialization of individual primitives in different parts of the state space and would require a similar competitive pressure as in Goyal et al. [26] to get weights over activation of individual primitives based on the primitive encoder's information content. Given models generalize a lot better than individual policies across different tasks, our hypothesis is that this would be more effective than the direct decomposition of policies. We aim to pursue this empirical enquiry in future work.

Acknowledgements We are thankful to the anonymous reviewers and everyone at the Stanford Intelligent Systems Laboratory for useful comments and suggestions. This work is supported in part by DARPA under Agreement Number D17AP00032. The content is solely the responsibility of the authors and does not necessarily represent the official views of DARPA. We are also grateful for the support from Google Cloud in scaling our experiments.

References

1. Abel, D., Hershkowitz, D. E., & Littman, M. L. (2016). Near optimal behavior via approximate state abstraction. In *International conference on machine learning (ICML)* (pp. 2915–2923).
2. Abel, D., Arumugam, D., Lehnert, L., & Littman, M. L. (2017). Toward good abstractions for lifelong learning. In *Proceedings of the NIPS workshop on hierarchical reinforcement learning*.

3. Al-Shedivat, M., Bansal, T., Burda, Y., Sutskever, I., Mordatch, I., & Abbeel. P, (2018). Continuous adaptation via meta-learning in nonstationary and competitive environments. In *International conference on learning representations (ICLR)*.
4. Anand, A., Grover, A., Singla, P., et al. (2015). ASAP-UCT: Abstraction of state-action pairs in UCT. In *International joint conference on artificial intelligence (IJCAI)*.
5. Andre, D., & Russell, S. J. (2002). State abstraction for programmable reinforcement learning agents. In *AAAI conference on artificial intelligence (AAAI)* (pp. 119–125).
6. Bacon, P., Harb, J., & Precup, D. (2017). The option-critic architecture. In *AAAI conference on artificial intelligence (AAAI)* (pp. 1726–1734).
7. Baird, L. C. (1994). Reinforcement learning in continuous time: Advantage updating. *IEEE International Conference on Neural Networks (ICNN)*, *4*, 2448–2453.
8. Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., van Hasselt, H. P., & Silver, D. (2017). Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems (NeurIPS)* (pp. 4055–4065).
9. Bertsekas, D. P., & Castanon, D. A. (1989). Adaptive aggregation methods for infinite horizon dynamic programming. *IEEE Transactions on Automatic Control*, *34*(6), 589–598.
10. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI gym. *CoRR*. arXiv:1606.01540.
11. Brunskill, E., & Li, L. (2014). PAC-inspired option discovery in lifelong reinforcement learning. In *International conference on machine learning (ICML)* (pp. 316–324).
12. Cobo, L. C., Isbell Jr, C. L., & Thomaz, A. L. (2012). Automatic task decomposition and state abstraction from demonstration. In *International conference on autonomous agents and multiagent systems (AAMAS)* (pp. 483–490). International Foundation for Autonomous Agents and Multiagent Systems.
13. Daniel, C., Neumann, G., Kroemer, O., & Peters, J. (2016). Hierarchical relative entropy policy search. *Journal of Machine Learning Research*, *17*(1), 3190–3239.
14. Dayan, P. (1993). Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, *5*(4), 613–624.
15. Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *IEEE computer society conference on computer vision and pattern recognition (CVPR)* (pp. 248–255).
16. Denis, N., & Fraser, M. (2019). Options in multi-task reinforcement learning: Transfer via reflection. In *Canadian conference on artificial intelligence* (pp. 225–237). Springer.
17. Eysenbach, B., Gupta, A., Ibarz, J., & Levine, S. (2018). Diversity is all you need: Learning skills without a reward function. In *International conference on learning representations (ICLR)*.
18. Finn, C., Abbeel, P., & Levine, S. (2017a). Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning (ICML)* (pp. 1126–1135).
19. Finn, C., Yu, T., Zhang, T., Abbeel, P., & Levine, S. (2017b). One-shot visual imitation learning via meta-learning. In *Conference on robot learning* (pp. 357–368).
20. Florensa, C., Duan, Y., & Abbeel, P. (2016). Stochastic neural networks for hierarchical reinforcement learning. In *International conference on learning representations (ICLR)*.
21. Frans, K., Ho, J., Chen, X., Abbeel, P., & Schulman, J. (2018). Meta learning shared hierarchies. In *International conference on learning representations (ICLR)*.
22. Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Ali Eslami, S. M., & Teh, Y. W. (2018). Neural processes. *CoRR*. arXiv:1807.01622.
23. Ge, L., Gao, J., Ngo, H., Li, K., & Zhang, A. (2014). On handling negative transfer and imbalanced distributions in multiple source transfer learning. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, *7*(4), 254–271.
24. Gershman, S. J. (2018). The successor representation: Its computational logic and neural substrates. *Journal of Neuroscience*, *38*(33), 7193–7200.
25. Goyal, A., Islam, R., Strouse, D., Ahmed, Z., Larochelle, H., Botvinick, M., Levine, S., & Bengio, Y. (2019a). Transfer and exploration via the information bottleneck. In *International conference on learning representations (ICLR)*.
26. Goyal, A., Sodhani, S., Binas, J., Peng, X.B., Levine, S., & Bengio, Y. (2019b). Reinforcement learning with competitive ensembles of information-constrained primitives. *CoRR*. arXiv:1906.10667.
27. Grant, E., Finn, C., Levine, S., Darrell, T., & Griffiths, T. (2018). Recasting gradient-based meta-learning as hierarchical Bayes. *CoRR*. arXiv:1801.08930.

28. Guestrin, C., Koller, D., Gearhart, C., & Kanodia, N. (2003). Generalizing plans to new environments in relational MDPs. In *International joint conference on artificial intelligence (IJCAI)* (pp. 1003–1010). Morgan Kaufmann Publishers Inc.
29. Ha, D., & Schmidhuber, J. (2018). World models. *CoRR*. arXiv:1803.10122.
30. Hafez-Kolahi, H., & Kasaei, S. (2019). Information bottleneck and its applications in deep learning. *Information Systems and Telecommunication*, *3*(4), 119.
31. Harb, J., Bacon, P. L., Klissarov, M., & Precup, D. (2018). When waiting is not an option: Learning options with a deliberation cost. In *AAAI conference on artificial intelligence (AAAI)*.
32. Holland, G. Z., Talvitie, E., & Bowling, M. (2018). The effect of planning shape on dyna-style planning in high-dimensional state spaces. *CoRR*. arXiv:1806.01825.
33. Isele, D., & Cosgun, A. (2018). Selective experience replay for lifelong learning. In *AAAI conference on artificial intelligence (AAAI)*.
34. Isele, D., Rostami, M., & Eaton, E. (2016). Using task features for zero-shot knowledge transfer in lifelong learning. In *International joint conference on artificial intelligence (IJCAI)* (pp. 1620–1626).
35. Jain, A., Khetarpal, K., & Precup, D. (2018). Safe option-critic: Learning safety in the option-critic architecture. *CoRR*. arXiv:1807.08060.
36. Jong, N. K., & Stone, P. (2005). State abstraction discovery from irrelevant state variables. *International Joint Conference on Artificial Intelligence (IJCAI)*, *8*, 752–757.
37. Keller, G. B., Bonhoeffer, T., & Hübener, M. (2012). Sensorimotor mismatch signals in primary visual cortex of the behaving mouse. *Neuron*, *74*(5), 809–815. https://doi.org/10.1016/j.neuron.2012.03.040.
38. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, *114*(13), 3521–3526.
39. Kulkarni, T. D., Saeedi, A., Gautam, S., & Gershman, S. J. (2016). Deep successor reinforcement learning. *CoRR*. arXiv:1606.02396.
40. Leinweber, M., Ward, D. R., Sobczak, J. M., Attinger, A., & Keller, G. B. (2017). A sensorimotor circuit in mouse cortex for visual flow predictions. *Neuron*, *95*(6), 1420–1432. https://doi.org/10.1016/j.neuron.2017.08.036.
41. Li, L., Walsh, T. J., & Littman, M. L. (2006). Towards a unified theory of state abstraction for MDPs. In *ISAIM*.
42. Liu, M., Machado, M. C., Tesauro, G., & Campbell, M. (2017). The eigenoption-critic framework. *CoRR*. arXiv:1712.04065.
43. Machado, M. C., Rosenbaum, C., Guo, X., Liu, M., Tesauro, G., & Campbell, M. (2017). Eigenoption discovery through the deep successor representation. In *International conference on learning representations (ICLR)*.
44. Machado, M. C., Bellemare, M. G., & Bowling, M. (2018). Count-based exploration with the successor representation. *CoRR*. arXiv:1807.11622.
45. Masoudnia, S., & Ebrahimpour, R. (2014). Mixture of experts: A literature survey. *Artificial Intelligence Review*, *42*(2), 275–293.
46. McCloskey, M., & Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In G. H. Bower (Ed.), *Psychology of learning and motivation, Vol 24, pp 109–165*. Cambridge: Academic Press.
47. Mendelssohn, R. (1982). An iterative aggregation procedure for Markov decision processes. *Operations Research*, *30*(1), 62–73.
48. Neumann, G., Daniel, C., Paraschos, A., Kupcsik, A., & Peters, J. (2014). Learning modular policies for robotics. *Frontiers of Computational Neuroscience*, *8*(62), 1–32.
49. Nguyen-Tuong, D., & Peters, J. (2011). Model learning for robot control: A survey. *Cognitive Processing*, *12*(4), 319–340.
50. Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., & Wermter, S. (2019). Continual lifelong learning with neural networks: A review. *Neural Networks*, *113*, 54–71. https://doi.org/10.1016/j.neunet.2019.01.012.
51. Reyman, G., & van der Wal, J. (1988). Aggregation–disaggregation algorithms for discrete stochastic systems. In *DGOR/NSOR* (pp. 515–522). Springer, Berlin.
52. Rosenbaum, D., & Weiss, Y. (2015). The return of the gating network: Combining generative models and discriminative training in natural image priors. In *Advances in neural information processing systems (NeurIPS)* (pp. 2683–2691).
53. Rosenstein, M. T., Marx, Z., Kaelbling, L. P., & Dietterich, T. G. (2005). To transfer or not to transfer. In *NIPS 2005 workshop on transfer learning* (Vol. 898, p. 3).
54. Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., & Hadsell, R. (2016). Progressive neural networks. *CoRR*. arXiv:1606.04671.

55. Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2015). High-dimensional continuous control using generalized advantage estimation. *CoRR*. arXiv:1506.02438.
56. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *CoRR*. arXiv:1707.06347.
57. Shamwell, E., Nothwang, W., & Perlis, D. (2018). An embodied multi-sensor fusion approach to visual motion estimation using unsupervised deep networks. *Sensors*, *18*(5), 1427.
58. Sun, C., Shrivastava, A., Singh, S., & Gupta, A. (2017). Revisiting unreasonable effectiveness of data in deep learning era. In *IEEE international conference on computer vision (ICCV)* (pp. 843–852).
59. Sung, F., Zhang, L., Xiang, T., Hospedales, T., & Yang, Y. (2017). Learning to learn: Meta-critic networks for sample efficient learning. *CoRR*. arXiv:1706.09529.
60. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. Cambridge: MIT Press.
61. Sutton, R. S., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, *112*(1–2), 181–211.
62. Talvitie, E. (2017). Self-correcting models for model-based reinforcement learning. In *AAAI conference on artificial intelligence (AAAI)*.
63. Tanaka, F., & Yamamura, M. (2003). Multitask reinforcement learning on the distribution of MDPs. *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, *3*, 1108–1113.
64. Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., de Las Casas, D., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T. P., & Riedmiller, M. A. (2018). Deepmind control suite. *CoRR*. arXiv:1801.00690.
65. Teh, Y., Bapst, V., Czarnecki, W. M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N., & Pascanu, R. (2017). Distral: Robust multitask reinforcement learning. In *Advances in neural information processing systems (NeurIPS)* (pp. 4496–4506).
66. Tessler, C., Givony, S., Zahavy, T., Mankowitz, DJ., & Mannor, S. (2017). A deep hierarchical approach to lifelong learning in minecraft. In *AAAI conference on artificial intelligence (AAAI)*.
67. Thrun, S. (1995). A lifelong learning perspective for mobile robot control. In *Intelligent robots and systems* (pp. 201–214). Elsevier.
68. Thrun, S., & Pratt, L. (1998). Learning to learn: Introduction and overview. In S. Thrun & L. Pratt (Eds.), *Learning to learn* (pp. 3–17). Boston: Springer.
69. Tiwari, S., & Thomas, P. S. (2018). Natural option critic. *CoRR*. arXiv:1812.01488.
70. Todorov, E., Erez, T., & Tassa, Y. (2012). MuJoCo: A physics engine for model-based control. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 5026–5033). https://doi.org/10.1109/IROS.2012.6386109.
71. Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., & Kavukcuoglu, K. (2017). FeUdal networks for hierarchical reinforcement learning. In *International conference on machine learning (ICML)* (pp. 3540–3549).
72. Wilson, A., Fern, A., Ray, S., & Tadepalli, P. (2007). Multi-task reinforcement learning: A hierarchical Bayesian approach. In *International conference on machine learning (ICML)* (pp. 1015–1022).
73. Yang, Y., Caluwaerts, K., Iscen, A., Tan, J., & Finn, C. (2019). NoRML: No-reward meta learning. In *International conference on autonomous agents and multiagent systems (AAMAS)* (pp. 323–331). International Foundation for Autonomous Agents and Multiagent Systems.
74. Zhang, J., Springenberg, J. T., Boedecker, J., & Burgard, W. (2017). Deep reinforcement learning with successor features for navigation across similar environments. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 2371–2378).
75. Zhang, S., & Whiteson, S. (2019). DAC: The double actor-critic architecture for learning options. *CoRR*. arXiv:1904.12691.
76. Zhu, Y., Gordon, D., Kolve, E., Fox, D., Fei-Fei, L., Gupta, A., Mottaghi, R., & Farhadi, A. (2017). Visual semantic planning using deep successor representations. In *Proceedings of the IEEE international conference on computer vision* (pp. 483–492).