Model Primitive Hierarchical Lifelong Reinforcement Learning

Bohan Wu
Columbia University
bw2505@columbia.edu

Jayesh K. Gupta
Stanford University
jkg@cs.stanford.edu

Mykel J. Kochenderfer
Stanford University
mykel@stanford.edu

ABSTRACT

Learning interpretable and transferable subpolicies and performing task decomposition from a single, complex task is difficult. 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 paper presents a framework for using diverse suboptimal world models to decompose complex task solutions into simpler modular subpolicies. This framework performs automatic 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

ACM Reference Format:
Bohan Wu, Jayesh K. Gupta, and Mykel J. Kochenderfer. 2019. Model Primitive Hierarchical Lifelong Reinforcement Learning . In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 9 pages.

1 INTRODUCTION

In the lifelong learning setting, we want our agent to solve a *series* of related tasks drawn from some task distribution rather than a single, isolated task. Agents must be able to *transfer* knowledge gained in previous tasks to improve performance on future tasks. This setting is different from multi-task reinforcement learning [25, 27, 32] and various meta-reinforcement learning settings [7, 8], where the agent jointly trains on multiple task environments. Not only do such non-incremental settings make the problem of discovering common structures between tasks easier, they allow the methods to ignore the problem of catastrophic forgetting [16], which is the inability to solve previous tasks after learning to solve new tasks in a sequential learning setting.

Our work takes a step towards solutions for such incremental settings [28]. We draw on the idea of modularity [17]. While learning to perform a complex task, we force the agent to break its solution down into simpler subpolicies instead of learning a single monolithic policy. This decomposition allows our agent to rapidly learn another related task by transferring these subpolicies. We hypothesize that many complex tasks are heavily structured and hierarchical in nature. The likelihood of transfer of an agent's solution increases if it can capture such shared structure.

A key ingredient of our proposal is the idea of world models [10, 12, 14] — transition models that can predict future sensory data given the agent's current actions. The world however is complex, and learning models that are consistent enough to plan with is not only hard [24], but planning with such one-step models is suboptimal [11]. 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 subtask. The same model primitives are used to learn a gating controller to select, improve, adapt, and sequence the various subpolicies to solve a given task in a manner very similar to a mixture of experts framework [15].

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 [3, 5]. Even though successor representations decouple the state transitions from the rewards (representing the task or goals), the transitions learned are policy dependent and can only transfer across tasks with the same environment dynamics.

There are alternative approaches to learning hierarchical spatio-temporal decompositions from the rewards seen while interacting with the environment. These approaches include meta-learning algorithms like Meta-learning Shared Hierarchies (MLSH) [8], which require a multiplicity of pretrained subpolicies and joint training on related tasks. Other approaches include the option-critic architecture [1] that allows 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. [31], its learning often collapses to a single subpolicy. 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.

To summarize our contributions:

- Given diverse suboptimal world models, we propose a method to leverage them for task decomposition.
- We propose an architecture to jointly train decomposed subpolicies and a gating controller to solve a given task.
- We demonstrate the effectiveness of this approach at both single-task and lifelong learning in complex domains with high-dimensional observations and continuous actions.

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada.

2 PRELIMINARIES

We assume the standard reinforcement learning (RL) formulation: an *agent* interacts with an *environment* to maximize the expected reward [23]. 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.

Lifelong Reinforcement Learning: In a lifelong learning setting, the agent must interact with multiple tasks and successfully solve each of them. Adopting the framework from Brunskill and Li [4], 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. After solving the given MDP or after H timesteps, whichever occurs first, the agent resamples from D and repeats.

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 [29] 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 [19]. The second category learns constraints that can be superimposed when learning a new function [13].

A popular approach within the first category is to use what Thrun and Pratt [29] term as *recursive* functional decomposition. This approach assumes that solution to tasks 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. This scheme has been particularly effective in computer vision where early convolutional layers in deep convolutional networks trained on ImageNet [6, 22] 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 [29]. The assumption behind this decomposition is that each function f_i can be represented by a collection of functions $h_1, \ldots, h_m$,

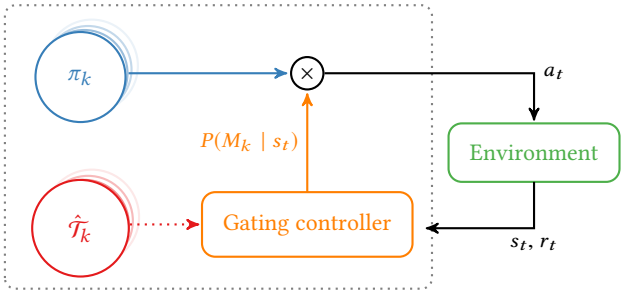


Figure 1: Diagram of MPHRL Architecture. Solid arrows are active during both learning and execution. Dotted arrows are active only during learning.

where $m \ll N$, and N is the number of tasks to learn. Our hypothesis is that this decomposition is much more effective and easier to learn in RL.

3 MODEL PRIMITIVE HIERARCHICAL RL

This section outlines the Model Primitive Hierarchical Reinforcement Learning (MPHRL) framework (Figure 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.

Suppose we have access to K model primitives: $\hat{\mathcal{T}}_k : \mathcal{S} \times \mathcal{A} \to \Pi(\mathcal{S})$. 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 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.

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. Instead, model primitives give rise to useful functional decompositions and allow subpolicies to be learned in a model-free way.

3.1.2 Gating Controller. Taking inspiration from the mixture-of-experts literature [15], 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 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.

Since the gating controller's goal is to choose the subpolicy whose corresponding model primitive makes the best prediction for a given transition, using Bayes' rule we can write:

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

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

The agent only has access to the current state s_t during execution. 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:

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

This is equivalent to:

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

Unfortunately, computing these integrals requires expensive Monte Carlo methods. However, we can use an approximate method to achieve the same objective with discriminative learning [18].

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 minimize the conditional cross entropy loss between $\mathbb{E}_{s_{t+1}, a_t \sim P(s_{t+1}, a_t)}\left[P(M_k \mid s_t, a_t, s_{t+1})\right]$ and $P_\phi(M_k \mid s_t)$ for all sampled transitions (s_t, a_t, s_{t+1}) in a rollout:

$$\underset{\phi}{\text{minimize}}\ \mathcal{L}^{GC} \quad (4)$$

where

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

This is equivalent to an implicit Monte Carlo integration to compute the marginal if $s_{t+1}, a_t \sim P(s_{t+1}, a_t)$. Although we cannot query or sample from $P(s_{t+1}, a_t)$ 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 sufficient for achieving task decomposition.

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

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

3.1.4 Decoupling Cross Entropy from Action Distribution. During a rollout, the agent samples as follows:

$$a_t \sim \pi(a_t \mid s_t) \quad (7)$$

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

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 difficult. We alleviate this issue by ignoring π_k, effectively treating it as a distribution independent of k. This transforms Eq. 1 into:

$$\hat{P}(M_k \mid s_t, a_t, s_{t+1}) \propto P(M_k \mid s_t)\hat{\mathcal{T}}(s_{t+1} \mid s_t, a_t, M_k) \quad (9)$$

3.2 Learning

Since the focus of this work is on difficult continuous 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:

$$\underset{\theta}{\text{maximize}}\ \mathcal{L}^{SP} = \mathbb{E}_{\rho_0, \pi_\theta}[\pi_\theta(a_t \mid s_t)Q_{\pi_\theta}(s_t, a_t)] \quad (10)$$

With baseline subtraction for variance reduction, this turns into [20]:

$$\underset{\theta}{\text{maximize}}\ \mathcal{L}^{PG} = \mathbb{E}_{\rho_0, \pi_\theta}[\pi_\theta(a_t \mid s_t)\hat{A}_t] \quad (11)$$

where $\hat{A}_t$ is an estimator of the advantage function [2].

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:

$$\hat{g}_k = \mathbb{E}_{\rho_0, \pi_{\theta_k}}\left[P_\phi(M_k \mid s_t)\nabla_{\theta_k}\log\pi_{\theta_k}(a_t \mid s_t)\hat{A}_t\right] \quad (12)$$

In practice, we use the Clipped PPO objective [21] 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:

$$\mathcal{L}^{BL} = \mathbb{E}\left[\left\|V_\psi - V_{\pi_\theta}\right\|^2\right] \quad (13)$$

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

```
Initialize P_φ, π_θ = {π_θ1, ..., π_θK}, V_ψ
while not converged do
    Rollout trajectories τ ~ π_θ,φ
    Compute advantage estimates Â_τ
    Optimize L^PG wrt θ_1, ..., θ_K
        with expectations taken over τ
    Optimize L^BL wrt ψ
        with expectations taken over τ
    Optimize 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_ψ
for Tasks $(\mathcal{R}_i, \mathcal{T}_i) \sim D$ **do**
  if RESET **then**
    Initialize P_ϕ, V_ψ
  while not converged **do**
    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 [30] ant navigating different mazes and a Stacker [26] arm picking up and placing 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 PPO algorithm as a baseline to compare against MPHRL. Transferring network weights empirically led to worse performance for standard PPO. Hence, we re-initialize its weights for every task. For fair comparison, we also shrink the hidden layer size of MPHRL's subpolicy networks from 64 to 16. We conduct each experiment across 5 different seeds. Error bars 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 controlled experiments with interpretable results, we hand-designed model primitives using the true next state provided by the environment simulator. 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:

$$\mu \sim \mathcal{N}(s_{t+1}, \sigma_k \Sigma) \tag{14}$$

$$\hat{\mathcal{T}}(s_{t+1} \mid s_t, a_t, M_k) = \mathcal{N}(\mu, \sigma_k \Sigma) \tag{15}$$

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.

Hyperparameter details are in Table 1, and our code is freely available at http://github.com/sisl/MPHRL.

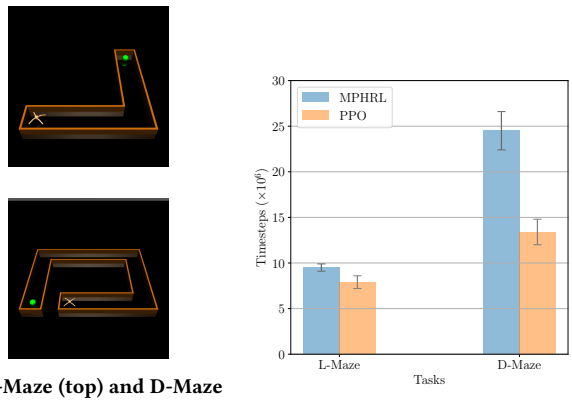
Table 1: Hyperparameters: MPHRL and baseline PPO

Category	Hyper-parameter	Value
Num. model primitives: Maze	L-Maze	2
	D-Maze	4
	Standard 10-Maze	4
	H-V Corridors	2
	Velocity	2
	Extra	5
Num. model primitives: 8-P&P[1]	Standard 8-P&P	12
	Box-only	2
	Action-only	6
Gating controller: Network	Hidden layers	2
	Hidden dimension	64
Gating controller: Base learning rate	Single / Source[2] (Maze)	1×10^{-3}
	Single / Source (8-P&P)	3×10^{-2}
	Target tasks	3×10^{-3}
Gating controller: Num. epoches / batch	Single / Source	1
	Target tasks	10
Baseline and model primitive networks[3]	Hidden layers	2
	Hidden dimension	64
	Base learning rate	3×10^{-4}
Subpolicy networks[4]	Hidden layers	2
	Hidden dimension (MPHRL)	16
	Hidden dimension (PPO)	64
	Base learning rate	3×10^{-4}
Optimization	Num. actors (Maze)	16
	Num. actors (8-P&P)	24
	Batch size / actor (Maze)	2048
	Batch size / actor (8-P&P)	1536
	Max. timesteps / task	3×10^7
	Minibatch size / actor	256
	Num. epoches / batch[5]	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

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 (Figure 2a) tasks require the ant to learn to walk and reach the green goal within a finite

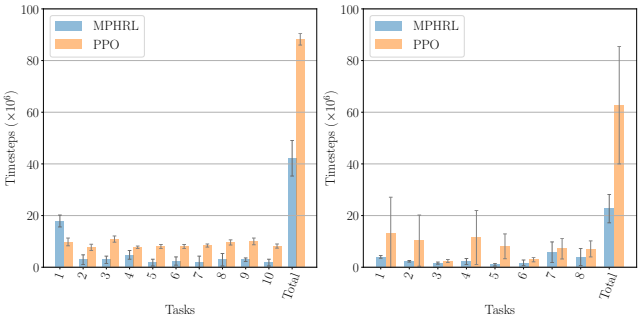
[1] 8-Pickup&Place.
[2] Single task refers to L-Maze and D-Maze; source and target tasks refer to the first task and all subsequent tasks in a lifelong learning taskset, respectively.
[3] Baseline network hyperparameters apply to both MPHRL and baseline PPO; model primitive networks are for experiments with learned model primitives only.
[4] The baseline PPO has no subpolicies, so the subpolicy network *is* the policy network.
[5] Baseline and subpolicy networks only.



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

(b) Performance

Figure 2: Single-task learning



(a) 10-Maze (b) 8-Pickup&Place

Figure 3: MPHRL vs. PPO for lifelong learning

horizon. For both tasks, both the goal and the initial ant locations are fixed. 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, $\sigma = 0.5$. The observation space includes the standard joint angles and positions, lidar information tracking distances from walls on each side, and the Manhattan distance to the goal. Figure 2b shows the experimental results on these environments. Notice that using model primitives can make the learning problem more difficult and increase the sample complexity on a single task. This is expected, since we are forcing the agent to decompose the solution, which could be unnecessary for easy tasks. However, we will observe in the following section that this decomposition can lead to remarkable improvements in transfer performance during lifelong learning.

4.2 Lifelong Learning

To evaluate our framework's performance at lifelong learning, we introduce two tasksets.

4.2.1 10-Maze. To evaluate MPHRL's performance in lifelong learning, we generate a family of 10 random mazes for the MuJoCo Ant environment, referred to as the 10-Maze taskset (Figure 4) hereafter. The goal, the observation space, the Gaussian noise models, and the model primitives remain the same as in D-Maze. The agent has a maximum of 3×10^7 timesteps to reach 80% success rate in each of the 10 tasks. As shown in Figure 3a, MPHRL requires nearly double the number of timesteps to learn the decomposed subpolicies in the first task. However, this cost gets heavily 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 [26] to create the 8-Pickup&Place taskset. As shown in Figure 5, 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 positions, box and goal locations, their relative distances to each other, and the current stage of the task encoded as one-hot vectors. The agent has access to six model primitives for each box that specialize in reaching above, lowering to, grasping, picking up, carrying, and dropping a certain box. Similar to 10-Maze, model primitives have σ of 0 within their specialized stages and σ of 0.5 otherwise. Figure 3b shows MPHRL's experimental performance by learning twelve useful subpolicies 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.

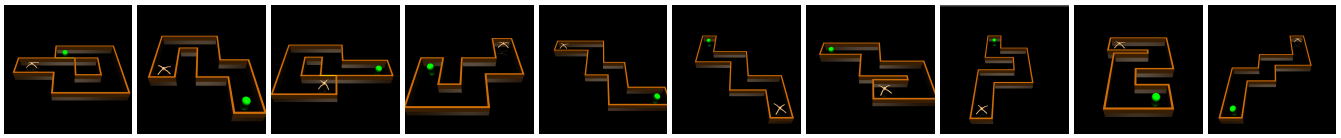
4.3 Ablation

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 the task decomposition even when the model primitives are quite noisy or when the source task does not cover all "cases"?
(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?
(6) Is MPHRL's gain in sample efficiency a result of hand-crafted model primitives and how does it perform with actual learned model primitives?

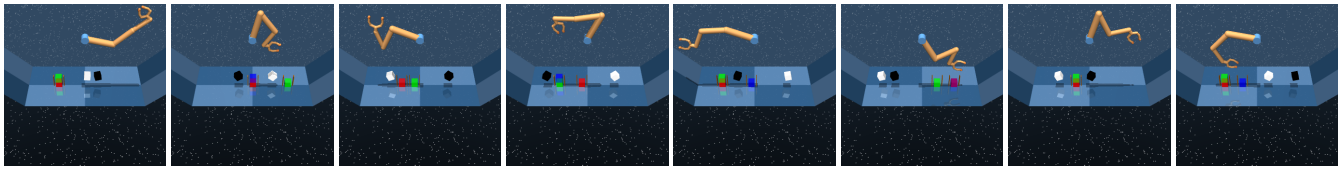
4.3.1 Model Noise. MPHRL has the ability to decompose the solution even given bad model primitives. Since the learning is done model-free, these suboptimal model primitives should not strongly affect the learning performance so long as they remain sufficiently distinct. To investigate the limitations to this claim, we conduct five 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 models with limited distinction
(b) 0.5 and 1.0: good models with reasonable distinction



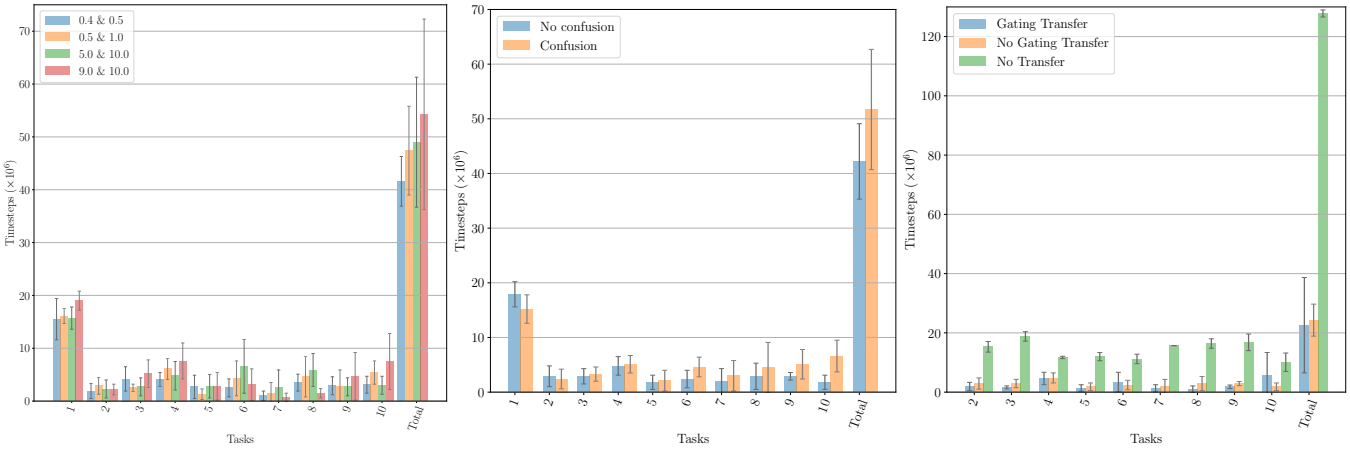
(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 4: 10-Maze lifelong learning taskset



(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)
B1 → T1 B2 → T2	B2 → T1 B1 → T2 B1 → T3	B2 → T1 B2 → T2	B1 → T1 B1 → T2 B2 → T3	B1 → T1 B2 → T2 B2 → T3	B1 → T1 B1 → T2 B1 → T3	B2 → T1 B1 → T2	B2 → T1 B1 → T2 B1 → T3

Figure 5: 8-Pickup&Place lifelong learning taskset. B1 and B2 refer to Box1 (black) and Box2 (white); T1, T2, and T3 refer to Target 1 (red), Target 2 (green), and Target 3 (blue)



(a) Effect of noisy model primitives

(b) Effect of model primitive confusion at the corners

(c) Effect of gating controller and subpolicy transfer (target tasks only)

Figure 6: 10-Maze: MPHRL ablation

Table 2: 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 of 80% (10-Maze) and 75% (8-Pickup&Place) ($\times 10^6$)										
Taskset	Model Primitives	1	2	3	4	5	6	7	8	9	10	Total
10-Maze	Extra	21.8±1.5	4.3±0.8	3.5±0.9	5.2±1.2	0.7±0.5	3.6±1.8	1.2±1.1	1.5±0.9	3.1±0.7	5.8±3.3	50.7±4.3
10-Maze	H-V Corridors	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-Pickup&Place	Action-only	5.1±0.6	4.8±1.8	16.8±12.6	26.9±6.9	N/A						
8-Pickup&Place	Box-only	18.6±10.9	N/A									

Table 3: 10-Maze: effect of experience

	Timesteps to reach 80% average success rate ($\times 10^6$)								
Experience	1	2	3	4	5	6	7	8	9
10 tasks	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 tasks	3.1±0.4	2.0±0.2	3.5±0.7	2.2±0.7	1.8±0.3				

(c) 5.0 and 10.0: bad models with reasonable distinction
(d) 9.0 and 10.0: bad models with limited distinction
(e) 0.5 and 0.5: good models with no distinction

Shown in Figure 6a, while (a), (b), (c), and (d) exhibit limited degradation in performance, (d) experiences the most performance degradation on average. On the other hand, in (e) MPHRL took 22.0±4.6 million timesteps to solve the 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. In summary, MPHRL is robust against bad model primitives so long as the they maintain some relative distinction. Similar observations hold true for the 8-Pickup&Place taskset where noise models with distinctive models with large noise of $\sigma = 5$ and $\sigma = 20$ show little deterioration in performance, taking 15.8 ± 5.5 million timesteps to reach 75% average success rate.

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 $\sigma = 0.5$. Figure 6b shows the performance for model primitive confusion against the standard set of model primitives with no confusion. We observe that despite some performance degradation, MPHRL continues to outperform the PPO baseline.

4.3.3 Model Diversity. Having tested MPHRL against noises, we experimented with undesirable model primitives for 10-Maze:

(a) *Extra*: a fifth model primitive that specializes in states where the ant is moving horizontally;
(b) *H-V corridors*: 2 model primitives specializing in horizontal (E, W) and vertical (N, S) corridors respectively;
(c) *Velocity*: 2 model primitives specializing in states where the ant is moving horizontally or vertically;

and for 8-Pickup&Place:

(a) *Box-only*: 2 model primitives for all actions on 2 boxes;
(b) *Action-only*: 6 model primitives for 6 actions performed on boxes: reach above, lower to, grasp, pick up, carry, and drop.

Table 2 shows MPHRL is susceptible to performance degradation given undesirable sets of model primitives. However, MPHRL still outperforms baseline PPO 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.

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. Ideally, they 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, and preferably better, than had it been exposed to T only.

In this experiment, we restore the subpolicy checkpoints after solving the 10 tasks and evaluate MPHRL's learning performance for the first 9 tasks. Similarly, we restore the subpolicy checkpoints after solving 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 3. Subpolicies trained sequentially on 6 or 10 tasks quickly relearn the required behavior for all previously seen tasks, implying no catastrophic forgetting. Moreover, if we compare the 10-task result to the 6-task result, we see remarkable improvements at transfer. This implies negative transfer is limited with this approach.

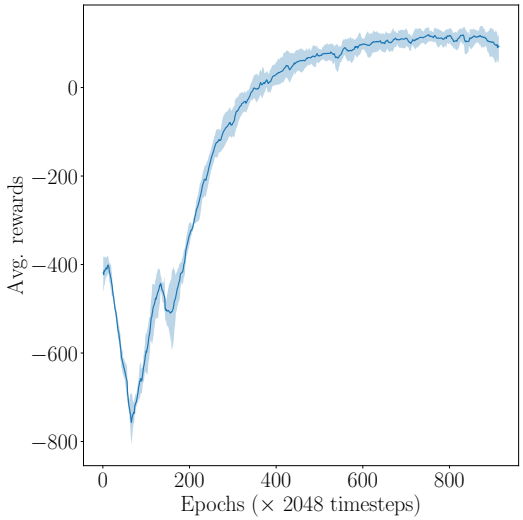


Figure 7: Average rewards of MPHRL when using an oracle gating controller. The reward threshold for reaching 80% success rate for the first task is approximately 800.

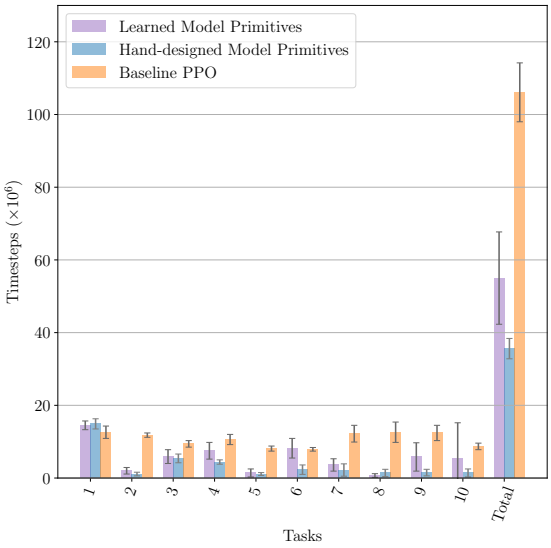


Figure 8: 10-Maze-v2: partial decomposition and learned model primitives. Success threshold is at 70%.

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, Figure 7 shows the reward curves for an experiment where the gating controller is already perfectly known. This setup is unable to

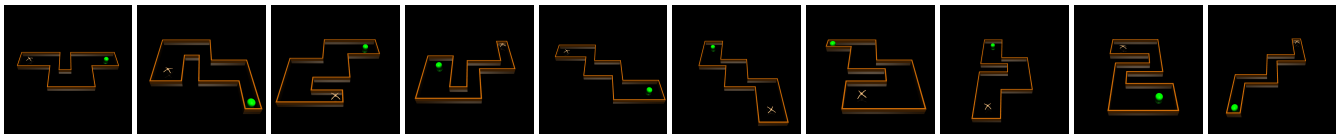


Figure 9: 10-Maze-v2 lifelong learning taskset

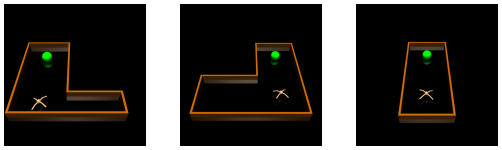


Figure 10: Three corridor environments for learning the "N" model primitive

learn any 10-Maze task. 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 policy needs to be learned from scratch, making the ant's survival rate very low. 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. In contrast, MPHRL's natural curriculum for gradual specialization allows multiple subpolicies to learn the basic skills for survival initially.

4.3.6 Partial Decomposition. To confirm that the ordering of tasks does not significantly affect MPHRL's performance, we modified 10-Maze to create the 10-Maze-v2 taskset (Figure 9), in which the source task does not allow for complete decomposition into all useful subpolicies for the subsequent tasks. Again, we observe large improvement in sample efficiency over standard PPO (Figure 8).

4.3.7 Learned Model Primitives. This paper focuses on evaluating suboptimal models for task decomposition in controlled experiments using hand-designed model primitives. Here, we show one way to obtain each model primitive for 10-Maze-v2 using three corridor environments demonstrated in Figure 10. Concretely, we parameterize each model primitive using a multivariate Gaussian distribution. We learn the mean of this distribution via a multilayer perceptron using a weighted mean square error in dynamics prediction as the loss. The standard deviation is still derived from the empirical covariance Σ as described earlier. Even though the diversity in these learned model primitives is much more difficult to quantify and control, their sample efficiency substantially outperforms standard PPO and slightly underperforms hand-designed model primitives with 0 and 0.5 model noises (Figure 8).

4.3.8 Gating Controller Transfer. To explore factors that lead to negative transfer, we tested MPHRL without re-initializing the gating controller in target tasks, as shown in Figure 6c. Although the mean sample efficiency remains stable, its standard deviation increases dramatically, indicating volatility due to negative transfer.

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. As shown in Figure 6c, sample complexity more than quintuples when subpolicies are re-initialized (in green).

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 4). 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.

Table 4: 10-Maze: effect of coupling between cross entropy and action distribution

	Timesteps to reach 80% average success rate ($\times 10^6$)				
Task	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

5 CONCLUSIONS

We showed how imperfect world models can be used to decompose a complex task into simpler ones. 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 transfer to a variety of related tasks, reducing the overall sample complexity required to learn complex behaviors. Our experiments showed that such structured decomposition avoids negative transfer and catastrophic interference, a major concern for lifelong learning systems.

Our approach does not require access to accurate world models. Neither does it need a well-designed task distribution or the incremental introduction of individual tasks. So long as the set of model primitives are useful across the task distribution, MPHRL is robust to other imperfections.

Nevertheless, learning useful and diverse model primitives, subpolicies and task decomposition *all simultaneously* is left for future work. The recently introduced Neural Processes [9] can potentially be an efficient approach to build upon.

ACKNOWLEDGMENTS

We are thankful to Kunal Menda and everyone at SISL 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] Pierre-Luc Bacon, Jean Harb, and Doina Precup. 2017. The Option-critic Architecture. In *AAAI Conference on Artificial Intelligence (AAAI)*. 1726–1734.

[2] L C Baird. 1994. Reinforcement Learning in Continuous Time: Advantage Updating. In *IEEE International Conference on Neural Networks (ICNN)*, Vol. 4. 2448–2453.

[3] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. 2017. Successor Features for Transfer in Reinforcement Learning. In *Advances in Neural Information Processing Systems (NIPS)*. 4055–4065.

[4] Emma Brunskill and Lihong Li. 2014. PAC-inspired Option Discovery in Lifelong Reinforcement Learning. In *International Conference on Machine Learning (ICML)*. 316–324.

[5] Peter Dayan. 1993. Improving Generalization for Temporal Difference Learning: The Successor Representation. *Neural Computation* 5, 4 (July 1993), 613–624.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A Large-scale Hierarchical Image Database. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 248–255.

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *International Conference on Machine Learning (ICML)*. 1126–1135.

[8] Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. 2018. Meta Learning Shared Hierarchies. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=SyX0IeWAW

[9] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, S M Ali Eslami, and Yee Whye Teh. 2018. Neural Processes. *CoRR* abs/1807.01622 (July 2018). arXiv:cs.LG/1807.01622

[10] D. Ha and J. Schmidhuber. 2018. World Models. *CoRR* abs/1803.10122 (2018). arXiv:cs.AI/1803.10122 https://worldmodels.github.io

[11] G. Zacharias Holland, Erik Talvitie, and Michael Bowling. 2018. The Effect of Planning Shape on Dyna-style Planning in High-dimensional State Spaces. *CoRR* abs/1806.01825 (June 2018). arXiv:cs.AI/1806.01825

[12] Georg B. Keller, Tobias Bonhoeffer, and Mark Hübener. 2012. Sensorimotor Mismatch Signals in Primary Visual Cortex of the Behaving Mouse. *Neuron* 74, 5 (2012), 809–815. https://doi.org/10.1016/j.neuron.2012.03.040

[13] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming Catastrophic Forgetting in Neural Networks. *Proceedings of the National Academy of Sciences* 114, 13 (2017), 3521–3526.

[14] Marcus Leinweber, Daniel R. Ward, Jan M. Sobczak, Alexander Attinger, and Georg B. Keller. 2017. A Sensorimotor Circuit in Mouse Cortex for Visual Flow Predictions. *Neuron* 95, 6 (2017), 1420–1432. https://doi.org/10.1016/j.neuron.2017.08.036

[15] Saeed Masoudnia and Reza Ebrahimpour. 2014. Mixture of Experts: A Literature Survey. *Artificial Intelligence Review* 42, 2 (Aug. 2014), 275–293.

[16] Michael McCloskey and Neal J Cohen. 1989. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. In *Psychology of Learning and Motivation*, Gordon H Bower (Ed.). Vol. 24. Academic Press, 109–165.

[17] Gerhard Neumann, Christian Daniel, Alexandros Paraschos, Andras Kupcsik, and Jan Peters. 2014. Learning Modular Policies for Robotics. *Frontiers of Computational Neuroscience* 8, 62 (June 2014), 1–32.

[18] Dan Rosenbaum and Yair Weiss. 2015. The Return of the Gating Network: Combining Generative Models and Discriminative Training in Natural Image Priors. In *Advances in Neural Information Processing Systems (NIPS)*. 2683–2691.

[19] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive Neural Networks. *CoRR* abs/1606.04671 (June 2016). arXiv:cs.LG/1606.04671

[20] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-dimensional Continuous Control Using Generalized Advantage Estimation. *CoRR* abs/1506.02438 (June 2015). arXiv:cs.LG/1506.02438

[21] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *CoRR* abs/1707.06347 (2017). arXiv:1707.06347

[22] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. 2017. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. In *IEEE International Conference on Computer Vision (ICCV)*. 843–852.

[23] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An Introduction*. MIT Press.

[24] Eric Talvitie. 2017. Self-Correcting Models for Model-Based Reinforcement Learning. In *AAAI Conference on Artificial Intelligence (AAAI)*.

[25] Fumihide Tanaka and Masayuki Yamamura. 2003. Multitask Reinforcement Learning on the Distribution of MDPs. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, Vol. 3. 1108–1113.

[26] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy P. Lillicrap, and Martin A. Riedmiller. 2018. DeepMind Control Suite. *CoRR* abs/1801.00690 (2018). arXiv:1801.00690

[27] Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. 2017. Distral: Robust Multitask Reinforcement Learning. In *Advances in Neural Information Processing Systems (NIPS)*. 4496–4506.

[28] Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J Mankowitz, and Shie Mannor. 2017. A Deep Hierarchical Approach to Lifelong Learning in Minecraft. In *AAAI Conference on Artificial Intelligence (AAAI)*.

[29] Sebastian Thrun and Lorien Pratt. 1998. Learning to Learn: Introduction and Overview. In *Learning to Learn*, Sebastian Thrun and Lorien Pratt (Eds.). Springer, Boston, MA, 3–17. https://doi.org/10.1007/978-1-4615-5529-2_1

[30] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. MuJoCo: A Physics Engine for Model-based Control.. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 5026–5033. https://doi.org/10.1109/IROS.2012.6386109

[31] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. 2017. FeUdal Networks for Hierarchical Reinforcement Learning. In *International Conference on Machine Learning (ICML)*. 3540–3549.

[32] Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. 2007. Multi-task Reinforcement Learning: A Hierarchical Bayesian Approach. In *International Conference on Machine Learning (ICML)*. 1015–1022.